НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
Теплоенергетичний факультет

Кафедра автоматизації проектування енергетичних процесів і систем

До захисту допущено:

Завідувач кафедри

_____ Олександр Коваль

«___»_____2020 р.

Дипломна робота

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Програмне забезпечення веб-технологій та мобільних пристроїв»

спеціальності 121 «Інженерія програмного забезпечення»

на тему: «Інструментальні засоби для тестування знань по різним дисциплінам з програмною генерацією питань»

Виконав
студент IV курсу, групи ТІ-62
Заїчко Олексій Павлович _____

Керівник:
доцент, к.т.н
Гагарін Олександр Олександрович _____

Рецензент:
к.т.н.,доцент
Степанець Олександр Васильович _____

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент (-ка) _____

Київ – 2020 року

Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет теплоенергетичний

Кафедра автоматизації проектування енергетичних процесів і систем

Рівень вищої освіти перший рівень

Напрям підготовки: 121 Інженерія програмного забезпечення

Спеціалізація: Програмне забезпечення веб-технологій та мобільних пристроїв

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____Олександр Коваль
(підпис)
”___”_____2020р.

ЗАВДАННЯ

на дипломну роботу студенту

_____Заїчку Олексію Павловичу_____
(прізвище, ім’я, по батькові)

1. Тема роботи: “Інструментальні засоби для тестування знань по різним дисциплінам з програмною генерацією питань”
керівник роботи __Гагарін Олександр Олександрович_____
(прізвище, ім’я, по батькові науковий ступінь, вчене звання)

затверджена наказом вищого навчального закладу від ”25” травня 2020р. № **1168-с**

2. Строк подання студентом роботи _____
3. Вихідні дані до роботи Проект Flutter, REST API
_____
_____

4.Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити проаналізувати існуючі програмні рішення для оцінювання знань в умовах дистанційної освіти, створити архітектуру і алгоритм системи, розробити програмне рішення і інтерфейс користувача_____
_____

5. Перелік ілюстративного матеріалу
схеми архітектури додатку, діаграма класів, діаграма структури системи, зразки розробленого інтерфейсу програми

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання ”___”_____201___ р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітки
1.	Затвердження теми роботи	16.10.2019	
2.	Вивчення та аналіз задачі	05.11.2019	
3.	Розробка архітектури та загальної структури системи	29.11.2019	
4.	Розробка структур окремих підсистем	16.01.2020	
5.	Програмна реалізація системи	13.04.2020	
6.	Оформлення пояснювальної записки	15.05.2020	
7.	Захист програмного продукту	09.06.2020	
8.	Передзахист	09.06.2020	
9.	Захист	16.06.2020	

Студент _____ _____
(підпис) (прізвище та ініціали,)

Керівник роботи _____ _
(підпис) (прізвище та ініціали,)

АНОТАЦІЯ

Мета роботи – створення гібридного застосунку, який являє собою клієнтську частину системи тестування знань «Test of Knowledge». Для реалізації інтерфейсу користувача і логіки програми була вибрана технологія «Flutter». Тестові питання генеруються програмно на серверній частині системи з використанням понятійно-тезисних моделей. Для спілкування між частинами системи використовуються HTTP-запити.

В результаті виконання роботи був досліджений поточний стан дистанційної освіти, підходи до складання тестових запитань. Було досліджено переваги і недоліки гібридних технологій. Було обговорено популярні архітектурні підходи для розробки застосунку.

Однією з особливостей даної системи є використання технології «Flutter», яка з’явилася порівняно недавно і дозволяє розробляти застосунки для смартфонів і браузерів з використанням єдиної кодової бази.

Функціональні можливості програми дозволяє користувачеві проходити тести під своїм іменем і групою. Інтерфейс дозволяє вибрати бажану дисципліну для контролю знань і складність питань. Після проходження тесту користувачеві оголошується результат тестування і видається список посилань на теми, по яким були допущені помилки.

Ключові слова: дистанційне навчання, автоматизація тестування, розробка ПЗ, Flutter, нативні додатки, REST API, понятійно-тезисна модель, архітектура BLOC.

Обсяг звіту становить 48 сторінок 14 ілюстрацій, 3 додатки і 13 посилань.

ABSTRACT

The purpose of this work was to develop hybrid application which is client-side part of the knowledge test system “Test of Knowlegde”. “Flutter” technology has been used for development of business logic and user interface. Test question are generated programmatically on server-side of the system using conceptual-thesis model. Communication between client and server is done vie HTTP requests.

In result, information about current state of distant learning and ways to generate test tasks has been gathered. In addition, pros and cons of hybrid technologies have been analyzed.

One particular thing about the system is usage of recent “Flutter” technology which enables developers to make application for both smartphones and browsers using the exact same codebase. Different application architectures have been discussed.

Features of the application enables user to complete tests using their name and group name. UI allows selection of desired discipline and difficulty of test question. After completion of a test user is informed about result and given a list of links to resources about topics where student made mistakes.

Keywords: Distance learning, test automation, software development, Flutter, hybrid applications, REST API, concept-thesis model, BLoC architechture.

The note contains 48 pages with 14 figures 3 annexes, and 13 references.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

API (англ. Application Programming Interface) — прикладний програмний інтерфейс;

UI (англ.) – користувацький інтерфейс;

Фреймворк — заготовки, шаблони для програмної платформи, що визначають архітектуру програмної системи; програмне забезпечення, що полегшує розробку і об'єднання різних модулів програмного проекту;

ПТМ – Понятійно-тезисна модель.

AI (англ.) – штучний інтелект.

ВСТУП

Основною перевагою дистанційної освіти завжди була можливість для навчання людей, які не могли б отримати її іншим способом. З появою нових технологій з'явилась можливість отримати знання не тільки в межах вищого навчального закладу. Все, що зараз потрібно для отримання освіти це скринька електронної пошти. Ще однією можливістю дистанційної освіти, якої немає в очному варіанті – це індивідуальний підхід до кожного студента.

Традиційно дистанційне навчання застосовувалось лише в окремих випадках, наприклад неможливість відвідувати заняття очно, наприклад, працюючі люди, військові, люди, які не є громадянами, тощо. Популярність дистанційного навчання зростає не просто так. Університетам вигідно мати студентів, заради яких не потрібно будувати нові корпуси або проводити закупку обладнання. В той самий час студенти самі можуть вибирати де і коли здобувати освіту.

З набуттям популярності віддаленого навчального процесу постає проблема перевірки знана, набутих у процесі дистанційного навчання. На момент виконання роботи широкого вжитку набули такі сучасні технології як Інтернет, відеоконференції, застосунки для миттєвого обміну повідомленнями. Враховуюче те, що у переважної більшості студентів в наявності є як мінімум смартфон або портативний комп'ютер, що забезпечує вільний доступ до мережі, найбільш доцільною вбачається розробка програмного забезпечення для системи автоматизованого тестування, яка забезпечить підтримку як мобільних платформ, так і роботу з використанням інтернет-браузера. Застосування автоматизації в освіті дозволяє індивідуалізувати процес навчання, забезпечити самоконтроль знань і контроль з діагностикою помилок й зворотнім зв'язком. Саме тому в цій праці була використана програмна генерація питань. Серед підходів до генерації була використана понятійно-тезисна модель, яка описується в наступних розділах. Він був обраний через відносно менші витрати часу і можливість створення бази знань використовуючи навчальний матеріал.

Після дослідження доступних інструментів для розробки до заданих вимог найбільше відповідає технологія Flutter, яка активно розробляється компанією Google

і використовується в таких проєктах як Alibaba, Google Greentea, Google Ads, App tree, Tencent. Дана технологія є кросплатформеною, що дозволяє швидко створити мобільні і веб-застосунки, без необхідності розробки під кожну окремо взяту платформу. Програми написані на Flutter є нативними, тому їх на перший погляд не відрізнити від застосунків спеціально розроблених для IOS або Android. Для роботи використовується нова мова програмування Dart, яка є дуже схожою на мови C++ й JAVA з деякими додатками. В питанні швидкості розробки Flutter не має жодних проблем. Для цього в стандартній бібліотеці є велика кількість вже готових віджетів, які можуть виконувати найрізноманітніші функції. [1]

Записка містить 5 розділів.

У першому розділі описується постановка задачі розробки клієнтської частини системи тестування знань «Test of Knowledge»

У другому розділі описується поточний стан дистанційного навчання і аналіз використаного підходу генерації тестових завдань.

У третьому розділі вказуються основні засоби розробки даної системи.

У четвертому розділі дано опис реалізованого програмного продукту і його архітектури.

У п’ятому розділі описується робота користувача з застосунком, можливий сценарій роботи.

ПОСТАНОВКА ЗАДАЧІ РОЗРОБКИ КЛІЄНТСЬКОЇ ЧАСТИНИ СИСТЕМИ ТЕСТУВАННЯ ЗНАНЬ «TEST OF KNOWLEDGE»

Метою праці є розробка клієнтської частини системи для автоматизації тестування знань «Test of Knowledge» з програмною генерацією питань. Система надає можливість проходити тестування під своїм іменем і номером групи.

В ході роботи було визначені наступні задачі:

- Розглянути стан сучасної дистанційної освіти.
- Розглянути способи створення тестових завдань.
- Вибір підтримуваних платформ.
- Вибір технології для реалізації застосунку.
- Проектування і розробка застосунку.
- Створення дизайну графічного інтерфейсу користувача.

Реалізація функціональних можливостей застосунку були розбиті на підзадачі:

- Налаштувати Http-запити до серверної частини застосунку.
- Можливість вводу імені і групи на стартовій сторінці застосунку.
- Відображення доступних категорій для тестування і можливість вибору дисципліни.
- Можливість вибору складності дисципліни.
- Реалізація сторінки тесту з вибором однієї правильної відповіді.
- Реалізація сторінки тесту з вибором декільком правильних відповідей.
- Реалізація сторінки тесту з співставленням термінів до тверджень.
- Реалізація індикатора статусу проходження тесту (кількість питань в тесті, стан кожного питання окремо)
- Відображення результату тестування.
- Відображення списку посилань на інформаційні ресурси по темам, по яким були допущені помилки.

Вихідними даними системи є результати пройденого тестування і список посилань на освітні ресурси.

Вимоги до розроблюваного програмного засобу:

- Застосунок повинен підтримувати роботу як в браузері, так і на мобільних платформах IOS і Android.
- Клієнтська частина системи має складатися з екранів логіну, вибору тесту, тестових питань і екрану виводу результату.
- Вхідними даними системи є ввід користувача і дані з серверної частини застосунку.

2. ДИСТАНЦІЙНЕ НАВЧАННЯ

2.1 Поняття дистанційної освіти

Дистанційна освіта – це форма освіти, яка дозволяє студентам, які фізично не можуть приймати участь в учбовому процесі, отримувати доступ до навчальних ресурсів в Інтернеті. Її можна сприймати як інструмент, який можна використовувати у разі необхідності. Але ефективність від переходу з очного навчання сильно залежить підготовки учнів і викладачів, технологічних інструментів або від загального стану інфраструктури підтримки студентів. Це відрізняється від віртуальних шкіл і навчальних програм, які зазвичай мають виділену інфраструктуру для технічної підтримки учнів.

Віддалене навчання надає можливість для студентів і викладачів залишатися на зв’язку під час роботи з дому. Зазвичай дистанційна освіта використовується в непередбачуваних ситуаціях, які представляють загрозу для учасників навчального процесу.

Перехід до дистанційного навчання може утримати достатній рівень підготовки для фізичного повернення до навчального закладу, без необхідності виконання великої кількості підготовчої роботи для відповідності до запланованих контролів знань. Важливо зазначити, що зазвичай в дистанцій освіті викладачі і студенти не пристосовані до дистанцій між ними під час викладання, що може бути своєрідним викликом для обох сторін.

Інфраструктура дистанційної освіти безпосередньо впливає на успішність такого підходу. Дуже часто віддалене навчання застосовується в часи нестабільної ситуації, тому важливо намагатися не звалювати на плечі учнів і викладачів додаткові обов’язки. Найбільш ефективним даний підхід буде при наявності добре визначеної інфраструктури для підтримки виконання плану навчання. Однією з цілей даної роботи є створення програмного забезпечення для покращення інфраструктури і зробити віддалене навчання більш доступним і зручним.

Найбільш важливими елементами дистанційної освіти включають в себе час, комунікацію, технології і дизайн занять. Очевидно, що визначення цих частин спочатку вбереже від зволікань під час навчання.

Час – це перше що потрібно мати на увазі, тому що він встановлює очікування і межі як для студентів, так і для викладачів. Наприклад, коли розпочати заняття і скільки вони будуть тривати. Викладачам слід встановити період часу, коли вони будуть доступні для учнів, для швидкої відповіді на запитання. Студенти повинні бути проінформовані про час який вони повинні витрачати на виконання завдань або інших дій описанні в конкретному занятті. Якщо студенти повинні регулярно відмічатися, про це потрібно обов'язково наголосити.

Комунікація – ще один аспект який має бути визначений ще на початку занять. Кожен учень повинен точно знати як і коли має контактувати з викладачем. На даному етапі потрібно прийняти рішення де бути проходити спілкування, наприклад, електронна пошта або онлайн-чат, запасні варіанти на випадок виходу з ладу чи недоступності основного варіанту. До того ж потрібно встановити очікування на рахунок частоти таких контактів викладача з студентами.

Використовувані технології можуть сильно відрізнятися в залежності від ситуації. Якщо навчальні заклади надають апаратне забезпечення студентам для роботи з дому, то всі мають бути готовими. Але в переважній більшості видачі пристроїв не проводиться і приходиться розраховувати на те що є в наявності. У випадках неможливості роботи через Інтернет, навчальний заклад повинен визначити інші шляхи для отримання і здачі завдань. Навчальним закладам потрібно навести дуже детальну інформацію про доступ до онлайн-платформ, особливо, якщо вчителі і студенти раніше не користувалися такими системами. [2]

2.2 Існуючі системи дистанційного навчання

У процесі пошуку інформації, та аналізу існуючих рішень, було виявлено, що на даний момент подібні системи є досить складними у використанні, або реалізують поставлену задачу не в повному обсязі:

naurok.com.ua – система для дистанційного навчання. Інтерфейс користувача представлений у вигляді веб-сайту. Недоліками системи є відсутність мобільних додатків і обмежена кількість типів запитань. Не можна вибирати складність тесту. Питання і варіанти відповідей вводяться вручну.

green-way.com.ua – система підготовки до здачі екзамену для отримання водійських прав. Тести вводяться вручну і однотипні. Можливе недобросовісне проходження через фактичне запам’ятовування завдань.

miyklas.com.ua – питання для оцінки знань генеруються вручну, недоліки, що й у системах наведених вище.

Насправді існує багато систем-аналогів, але вони мають недоліки, які зустрічаються в системах, описаних вище. Але вони представляють собою комп’ютеризацію традиційного тестування, з усіма його недоліками, як високі витрати часу на створення достатньої бази завдань, погана оптимізація завдань у разі зміни навчального матеріалу і можливість недобросовісного виконання контролю знань.

2.3 Аналіз підходів генерації тестових завдань

Сьогодні багато праць в області комп’ютерного контролю знань фокусуються на надійності і валідності тестових завдань, створення матеріалів для них до цих пір залишається обов’язком викладача. Насправді, майже всі спроби запрограмувати формування тестових завдань все більше використовує штучний інтелект, але постає проблема формалізації знань та генерація тестів з її використанням. Основним підходом генерації засобів тестування є алгоритмізація і комп’ютеризація ручного тестування. Сенс такого способу полягає заміну паперової роботи на інформаційно-

комунікаційні технології, що надає такі переваги як автоматизоване формування тестів і їх перевірка. Неоднозначним аспектом цього є ручне створення завдань.

Якісні і зрозумілі завдання може розробити експерт. Питання валідності і надійності не є проблемою при використанні статичних тестів. Проблемою є високі витрати часу на сам процес створення тестових завдань. Додаткові труднощі приносить задачу захисту від недобросовісного виконання оцінювання, що передбачає створення великої кількості варіантів набору тестових завдань. Це потребує від розробника завдань знання і аналізу безпосередньо навчального матеріалу, який вказується як джерело знань для студентів, не говорячи вже про експертні знання в області.

Для інтенсифікації і зменшення складності підготовки програм для дистанційного навчання з використанням тестових частин, була запропонована деяка кількість підходів для програмної генерації тестових завдань. Одним з перспективних і порівняно нескладних в реалізації є підхід параметризованих тестів. Суть підходу полягає у поданні різним студентам шаблонного завдання, яке відрізнятиметься певними параметрами, які генеруються автоматично. Відповідь вводиться з клавіатури. Таким чином кожен студент отримує індивідуальне завдання, а система по певній формулі чи алгоритму, підставляючи параметри отримує вірну відповідь для подальшої перевірки відповіді, введеної студентом. Недоліком підходу є його вузька предметна спрямованість. Так параметризовані тести добре підходять для організації контролю практичних навичок в точних науках, а також програмуванні, проте не підходять для перевірки теоретичних знань, а також контролю в гуманітарних науках.

Деякого поширення в алгоритмізації перевірки засвоєння навчального матеріалу отримав спосіб застосування семантичних мереж для побудови завдань. [3,4]. Основним елементом таких мереж є тріади [3-5]: сутність 1 – відношення – сутність 2. Тест створюється за допомогою опускання однієї з ланок тріади і заданням питання про ланку, яка є відсутньою. Плюсом такого способу є можливість використання знань з предметної області для створення завдань. Мінусом ж те, що для складання повної семантичної мережі, яка могла б правильно відображати конкретну предметну область потрібні значні витрати. Іншою складністю є

лінгвістична невалідність і іноді, недоцільність згенерованих тестових запитань. Це означає, що на основі семантичної мережі часто трапляються питання про предметну область, для відповіді на які потрібні знання особливостей, які не становлять будь-якої педагогічної цінності у освітньому контексті. Такі проблеми є типовими для класичних моделей знань з використання AI, також їх називають проблемами всеосвіченості.

Понятійно-тезисна модель (ПТМ) формалізації дидактичного тексту розробляється на стику багатьох наукових галузей, серед яких наступні: інженерія знань як напрямок штучного інтелекту; педагогіка, а саме її розділ – дидактика, що розкриває правила викладання; інтерпретація (герменевтика, екзегетика), що вивчає правила тлумачення текстів [6, 7]; лінгвістика і її розділ семантика, що вивчають закономірності природної мови і проблеми, пов’язані зі змістом, значенням, і інтерпретацією лексичних одиниць; інструментом реалізації служать технології розробки Web-систем.

У навчальному процесі можна виділити декілька основних аспектів: передача інформації, спілкування між викладачами і студентами, тобто соціальна складова, практика і контроль набутих у процесі знань. Основна ціль дистанційного навчання – зменшення складності і покращення передачі знань від їх носія до студента. Класичним способом передачі знань в традиційній формі навчання є передача вербальним способом. Але у віддаленому навчанні, без викладача таким способом є текстова інформація, гіпертекст і мультимедіа. Мається на увазі що основним способом протоколом передачі знань є текстова інформація, що є головним в понятійно-тезисній моделі знань.

Основною сутністю в ПТМ є «поняття», це об’єкт про який в навчальному матеріалі містяться знання, які потрібні студенту. Для кожної предметної області виділяються свої поняття. Множина понять в ПТМ позначається наступним чином. (2.3.1)

$$C = \{c_1, \dots, c_{n1}\} \qquad (2.3.1)$$

Для подання знань використовуються інші структурні елементи – тези, які представляються собою відомість про поняття. Теза – це ознака, яка є істиною для

конкретного поняття. З лінгвістичної точки зору, це речення або декілька речень, з яких видалені підмети. Множина тез у програмі позначається так (2.3.2):

$$T = \{t_1, \dots, t_{n2}\} \qquad (2.3.2)$$

В сукупності наведені вище елементи складають з себе ПТ-елементи. Будь-яка теза відповідає лише одному поняттю. Його позначено відношенням (2.3.3):

$$CT: T \to C \qquad (2.3.3)$$

Поняття може відповідати довільній кількості тез (2.3.4):

$$TC: C \to 2^T \qquad (2.3.4)$$

Центральним джерелом знань є навчальний матеріал. З нього за допомогою маніпуляцій виводяться конкретні семантичні одиниці. Навчальні матеріали для використання в дистанційній освіті діляться на невеликі частини для кращого розуміння студентами. Такі частини називаються «кадрами», що є дуже важливими для використання в ПТМ, так як саме з ним виділяються семантичні елементи. Вони позначаються таким чином (2.3.5):

$$V = \{v_1, \dots, v_{n3}\} \qquad (2.3.5)$$

Елементи ПТМ виводяться з тексту навчального матеріалу. Вони формуються способом осмисленого читання безпосереднього тексту з нескладними маніпуляціями. Викладач виділяє з тексту семантичні елементи і додає їх в базу даних. Кожен фрагмент може бути джерелом довільної кількості тез, що задається наступним відображенням (2.3.6).

$$TV: V \to 2^T \qquad (2.3.6)$$

Кожна t_j, у свою чергу, стосується одного навчального фрагменту vi (2.3.7):

$$VT: T \to V \qquad (2.3.7)$$

Маючи на увазі те, що тези стосуються лише одного фрагменту матеріалу, з якого вони були створенні, і те що поняття можуть відноситись до багатьох навчальних фрагментів, зв'язок між матеріалом і поняттями влаштовується використовуючи тези.

Для створення різних типів завдань використовуються шаблони. Вони дозволяють доповнювати різні предметні області завданнями нових типів. Шаблони, які використовуються для тестових запитань задаються наступним відображенням (2.3.8):

$$TaskTempl: Task \rightarrow TTempl \qquad (2.3.8)$$

Спосіб побудови тестових завдань на основі понятійно-тезисних елементів описується шаблонами. Функція додавання нових шаблонів надає можливість удосконалення системи не лише на етапу проектування, але й після її розгортання і введення в експлуатацію для налаштування алгоритму створення завдань для різних дисциплін.

Генерація завдання тестового завдання відбувається в декілька етапів.

Першим етапом є вибір контрольної ПТ-пари. Вона вибирається випадково з предметної області тесту.

На другому етапі виконується пошук допустимих шаблонів завдань для конкретного типу завдання. В основі шаблону може стояти як поняття, так і теза. Спочатку знаходяться шаблони з основою-поняттям.

Третій етап – це вибір шаблону питання випадковим чином з списку знайдених.

На четвертому етапі проводиться пошук альтернативних відповідей. Відповідно до поточного шаблону вибираються ПТ-пари, які потенційно можуть виступити як альтернативні варіанти відповідей. Можливі варіанти обираються з множини понять або тез, в залежності від вибраного шаблону.

На останньому етапі відбувається візуалізація, з використання текстового шаблону. У спеціальне місце вставляється текст запитального елемента контрольної ПТ-пари.

ПТМ надає можливість знайти рішення задач несумісності завдань традиційного AI і систем інтелектуального навчання за допомогою використання з позиції обробки навчального тексту безпосередньо, формалізації знань, який є першоджерелом знань. Для побудови бази знань використовується техніка осмисленого читання навчальних матеріалів для пришвидшення побудови бази.

Такий підхід дозволяє реалізувати генератор тестових питань для системи дистанційного навчання, що забезпечує неповторність кожного тесту. Це зводить до мінімуму проблему недобросовісного тестування, яка властива незмінним завданням тестування, з невеликою базою питань. Крім цього витрати часу на створення бази є меншими, ніж при ручній генерації достатньої для повної семантичної мережі предметної області завдань.

На відміну від тестових завдань, що генеруються на основі семантичних мереж, завдання на основі ПТМ володіють кращою лексичною зрозумілістю, що позитивно впливає на якість контролю. Точний зв'язок семантичних даних з ділянками навчального матеріалу дозволяє використовувати різноманітні стратегії при формуванні і оцінюванні тестів.

ПТМ інтегрується з ієрархічно-мережевою моделлю освітнього контенту Tree-Net, що дозволяє будувати індивідуалізоване навчаюче середовище з такими функціями, як генерація індивідуальних тестових завдань і генерація індивідуального навчального курсу [8, 9].

Практична реалізація ПТМ-підходів і автоматизоване тестування здійснені у рамках відкритого порталу дистанційного навчання [10]. Програмна реалізація містить засоби поповнення ПТ-бази на основі асинхронного Web-інтерфейсу і модуль генерації і аналізу тестів на основі ПТМ.

Подальші дослідження повинні бути зосереджені на підвищенні якості тестових завдань і вдосконаленні методів і алгоритмів їх генерації. Перспективним тут є розширення базової класифікації тез і понять та формування на основі цієї класифікації нових шаблонів для завдань тесту. Ще одним перспективним напрямком є побудова моделі шаблона для тестового завдання відкритого типу.

3. ЗАСОБИ РОЗРОБКИ ПРОГРАМНОГО ПРОДУКТУ

3.1 Фреймворк для гібридних додатків Flutter

Flutter – безкоштовна open-source мобільна фреймворк для побудови інтерфейсу користувача від компанії Google, яка вийшла в травні 2017 року. Однією з переваг є можливість написання єдиної кодової бази для платформ IOS і Android, також недавно з'явилася підтримка Flutter Web для веб браузерів. Це означає наявність можливості використання одної мови програмування для декількох додатків.

Flutter складається з двох важливих частин:

- SDK (Software Development Kit) – колекція інструментів, які допомагають в розробці застосунків. Це включає інструменти для компіляції коду в нативний машинний код.
- Framework – колекція віджетів для багаторазового використання UI елементів, які можна використати в своєму застосунку.

Flutter – це молода, але багатообіцяюча платформа, яка вже привернула увагу достатньо великих компаній, які запустили свої додатки. Цікава дана платформа своєю простотою, порівняно з розробкою веб-застосунку і швидкість роботи на рівні нативних додатків. Висока продуктивність застосунків і швидкість розробки досягається за рахунок декількох технік:

- На відмінну від багатьох відомих сьогодні мобільних платформ, Flutter не використовує Javascript взагалі. В якості мови програмування для Flutter вибрали Dart, який компілюються в бінарний код, завдяки якому досягається швидкість виконання операцій, яку можна порівняти з нативними технології.
- Flutter не використовує нативні компоненти, тому нічого не потрібно розробляти для комунікації з ними. Замість цього інтерфейс будується як в ігрових програмах. Фон, текст, медіа-елементи, кнопки – весь рендер відбувається в Flutter. Навіть враховуючи це, «Hello World» застосунок займає зовсім не багато місця.
- Для побудови інтерфейсу користувача використовується декларативний підхід, що схоже на шлях ReactJs, на основі віджетів. Для ще більшого приросту до

швидкості роботи інтерфейсу віджети малюються за необхідністю – тільки якщо в моделі відбулися зміни (подібно до того як відбувається у світу Front-End).

- Також варто зазначити, що в фреймворк вбудований так званий Hot-reload, якого до сих пір не було в нативних платформах.

Структура проекту на Flutter складається з декількох директорій:

- Lib/ - По принципам pub (менеджер пакетів Dart), весь код проекту лежить в даній директорії.
- Pubspec.yml – в цей файл буде записані всі залежності застосунку, які потрібно встановити для його запуску.
- Test/ – Unit тести для додатку.
- Ios/ & android/ – директорії для кожної з платформ. Там вказуються які права потрібні для запуску застосунку (наприклад, доступ до локації, Bluetooth), а також те що є специфічним для кожної IOS і Android.

В Flutter все побудовано на віджетах: представлення, стилі з темами, навіть стан застосунку. Існує їх два типи – з станом і без. Віджети, у яких зміни моделі не призводять до перемалювання інтерфейсу називаються stateless. Якщо потрібно змінити, наприклад, текст потрібно створити новий екземпляр класу. На відміну від перших, stateful віджети реагують на зміни внутрішньої моделі.

3.2 Мова програмування Dart

Dart – це нова мова програмування, яка швидко набирає популярність, особливо з виходом фреймворку Flutter. Це гнучка мова підходить як і для простих програм, так і для повнофункціональних застосунків. Вона використовує об’єктно-орієнтований підхід і C-подібний синтаксис для більшої простоти і доступності.

Характеристика мови програмування Dart:

- Передбачається використання класів програмістами. На відмінну від мови програмування Javascript яка дає можливість вибору щодо використання класів. З Dart ви вимушені використовувати ООП. Є підтримка інтерфейсів, абстрактних класів, generics, mixins, статична типізація.

- В той час як більшість мов програмування мають статичну або динамічну типізацію. Є можливість декларувати типи змінних так використовувати змінні без явного вказання типу.
- Підтримується компіляція коду на Dart в Javascript для виконання в усіх сучасних браузерах.
- Також підтримується Isolates для потоків. Вони не розділяють пам'ять, а спілкування здійснюється за допомогою повідомлень.

Dart має коротку криву навчання, для того щоб набути всіх необхідних знань непотрібно витрачати дуже багату часу вивчаючи всі нюанси мови. Таку простоту мови забезпечує підтримка як і строгої, так і не строгої типізації. Це робить Dart простішим для тих, хто переходить з інших мов програмування. Синтаксис простий і його можна зрозуміти без особливих складностей. Хоча мова добре структурована так само, як і C, їй вдається перемогти останню з точки зору простоти.

Dart підтримує як і AOT, так і JIT компіляцію. Хоча дана функція доступна не на всіх фреймворках. Вони зазвичай використовуються в різних ситуаціях, наприклад, при компіляції нативних застосунків використовується AOT. JIT може бути дуже корисним під час етапу розробки, тому що вона дозволяє майже миттєво побачити зміни в коді застосунку. [11]

3.3 Архітектура BLoC

BLoC (Business Logic Components) – це архітектурний дизайн патерн. Початково, даний патерн був створений для можливості повторного використання коду незалежно від платформи: веб застосунки, мобільні застосунки, код серверу. Тому він був розроблений з прицілом на полегшення навантаження для розробників під час розробки програм для різних платформ з ідеєю повторного використання коду.

Для того щоб зрозуміти що таке BLoC потрібно знати що таке так звані Streams. У загальному, Stream – це безперервний потік або послідовність чого небудь. З технічного точки зору, Stream це ніщо інше як безперервний потік даних. Можна уявити собі конвеєрну стрічку. Конвеєр має два кінці: вхід і вихід. З входу речі подаються на конвеєр і після обробки ми отримуємо результат на виході.

Ключові поняття:

- Stream – конвеєр.
- StreamController – те, що контролює конвеєр.
- StreamTransformer – те, що обробляє вхідні дані.
- StreamBuilder – функція яка приймає потік як аргумент і надає нам Builder який викликається кожен раз коли з потоку виходять нові дані.
- Sink – поле, яке приймає вхідні дані до потоку.
- Stream – поле, яке надає доступ до вихідних даних потоку.

Одна з головних перешкод при програмуванні на Flutter це одночасна взаємодія з інтерфейсом користувача і з усім іншим, тому що у нас немає ніякої проміжної мови для дизайну UI як XML в Android, натомість в Flutter весь код пишеться в одному місці. З цим нам допомагає BLoC архітектура, яка ефективно усуває дані труднощі.

— BLoC не тільки сприяє повторному використанню коду, а також використовує потоки даних для контролю і поширення змін стану в Flutter.

— BLoC допомагає нам розділити бізнес логіку від користувацького інтерфейсу. UI компоненти повинні контролювати лише інтерфейс, але не логіку.

Як було обговорено вище BLoC використовує потоки даних, з цього можна зробити деякі висновки.

- Вхідні дані приймаються за допомогою поля sink.
- До вихідних даних можна звернутися за допомогою поля stream.
- Віджети відправляють події до BLoC за допомогою sink.
- BLoC сповіщає віджети про зміни через stream.
- Так як бізнес логіка застосунку і користувацький інтерфейс розділені, можлива її зміна без впливу на UI і навпаки.

Загальну схему роботи BLoC можна побачити на ілюстрації (Рисунок 3.3.1).

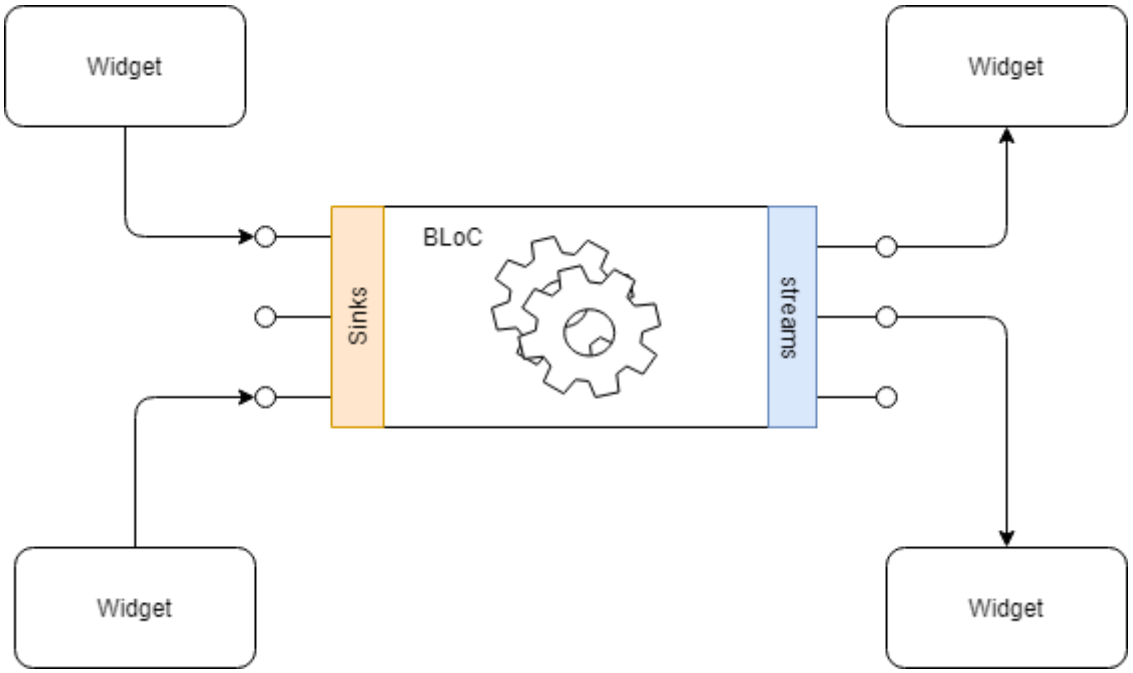

Рисунок 3.3.1 — Структура BLoC

Після того як було встановлено що із себе представляє даний архітектурний підхід, логічним виявляється питання навіщо для state management використовувати саме BLoC, коли в екосистемі платформи Flutter вже існують інші шляхи. Для визначення відповіді на це питання потрібно більш детально дослідити інші підходи і переконатися що BLoC є найефективнішим способом.

Першим способом контролю стану застосунку є метод setState(). Ця функція повідомляє фреймворк про зміну внутрішньої моделі об’єкту. Кожний раз, коли потрібно поміняти об’єкт стану, зміни потрібно робити в функцію, яка передається в setState() як аргумент.

Виклик setState() повідомляє фреймворк, що зміни у внутрішній моделі об’єкту можуть викликати мутації користувацького інтерфейсу в цьому піддереві. В свою чергу фреймворк запановує повторний build інтерфейсу застосунку. В даному підході існують дві основні проблеми. По перше, навіть проста зміна стану може викликати build всього дерева віджетів. По друге, це не вирішує проблему розділення коду для бізнес логіки і UI.

Другим способом є використання класу InheritedWidget. Наведений клас – спеціальний віджет, який визначає контекст в корені потрібного піддерева, з можливістю доступу до даного контексту в буде-якому дочірному віджеті. Основні

його функції це поширення змін вниз по дереву і оновлення віджетів при кожному build. Проблемами даного способу є те що стан InheritedWidget є final, що означає неможливість його зміни після створення. Також не вистачає функціональних можливостей для звільнення ресурсів і уникнення витоку даних.

Третім підходом є використання Scoped Model. Насправді це сторонній пакет, який підтримує Brain Egan. Він побудований на основі InheritedWidget, пропонуючи трохи кращий спосіб для доступу, запису і зміни стану застосунку. Це дозволяє нам передачу даних від батьківських до дочірніх віджетів, а також оновлювати всі віджети, які використовують модель коли вона змінюється. Для реєстрації змін стану в Scoped Model викликається метод notifyListeners(). На жаль, недоліком з ростом моделі буде важко орієнтуватися де саме потрібно викликати той самий notifyListeners().

З наведеного вище можна зробити висновок, що альтернативи мають деякі серйозні недоліки, в той час коли BLoC вирішує наведені проблеми.

Тепер постає питання коли потрібно використовувати даний підхід для максимальної ефективності. Для цього потрібно розібрати поняття локального і глобального стану. Для пояснення можна навести такий приклад: Форма входу в систему, де користувач має ввести ім’я і пароль і комунікація з API для валідації даних користувача перед спрацювання аунтефікації. В цьому випадку форму і її валідацію можна представити у вигляді локального стану, так як ця частина логіки відноситься лише до одного компоненту програми і більше не зустрічається. В той самий час, від комунікації застосунку з API залежить робота усього застосунку, тому ми відносимо це до глобального стану. Під час виконання роботи було визначено що BLoC найкраще використовувати для локального стану. У випадку з глобальним станом варто подивитися в сторону інших підходів, наприклад, Redux. [12]

3.4 Архітектура REST

REST – це архітектурний стиль або дизайн патерн для API. REST був визначений вченим по комп’ютерним наукам Roy Fielding. Перед тим як визначити

що робить API RESTful і яких правил й обмежень потрібно дотримуватися, розберемо два основних поняття.

Клієнт – це користувач або програма, яка використовує API. Це може бути, наприклад розробник, який використовує Twitter API для отримання і запису даних, створення нових твітів або виконання інших дій у своїй програмі. Застосунок буде використовувати Twitter API. Клієнтом також може бути веб браузер. Коли ви переходите на веб сайт Twitter, ваш браузер виступає у ролі клієнта, який викликає Twitter API і використовує отриману інформацію для відображення інформації на екрані.

Ресурс – будь-який об’єкт який може віддати API клієнту. В API Instagram, наприклад, ресурсом може бути користувач, фотографія або хештег. Кожен з них має унікальний ідентифікатор який може бути числом або масивом символів.

Застосунки з RESTful API описує себе через інформацію про доступ до його ресурсів. Також клієнту дозволяється маніпулювати ресурсами, наприклад, створити нові дані або поміняти існуючі. Для того щоб зробити своє API RESTful потрібно дотримуватися деяких обмежень при його програмуванні. Ці обмеження зробить розроблюване API простішим для використання і вивчення, для того щоб розробникам, які тільки починають користуватися API буде легше почати. [13]

Абревіатура REST розшифровується як REpresentational State Transfer. В перекладі на українську це означає, що при виконання запиту до RESTful, сервер відправить у відповідь клієнту представлення поточного стану ресурсів по яким відбувся запит. Наприклад, коли розробник викликає Instagram API для отримання даних про конкретного користувача, сервер поверне стан даного користувача, що може включати в себе ім’я, кількість публікацій і підписників, тощо.

Представлення може бути в JSON форматі, і для більшості API це є так. Іноді зустрічаються формати XML і HTML.

Те, що робить сервер при отримані запиту від клієнта залежить від двох характеристик.

- Ідентифікатор потрібних ресурсів. Це URL (Uniform Resource Locator) ресурсу, також відомий як ендпоінт.

- Операція яку потрібну виконати над цільовим ресурсом. Вона вказується в HTTP методі запиту. Найбільш поширені методи: GET, POST, PUT і DELETE.

Для того щоб API було RESTful, воно має дотримуватись таких правил.

- Uniform interface
- Client — server separation
- Stateless
- Layered system
- Cacheable
- Code-on-demand

Uniform interface складається з 4 частин:

— Запит до серверу має містити ідентифікатор ресурсу.

— Даних, які відправляє сервер повинно бути достатньо для того, щоб клієнт міг модифікувати їх.

— Кожен запит до API містить у собі всю необхідну інформацію для його обробки і кожна відповідь від серверу має достатньо інформації для розуміння її клієнтом.

— Гіпермедія як основа стану застосунку. Під застосунком розуміється веб сайт, який базується на сервері. Під гіпермедія розуміються гіперпосилання, прості посилання, які сервер включив у відповідь на запит. Це твердження означає, що сервер може інформувати клієнта про способи зміни стану веб застосунку. Якщо клієнт зробить запит на конкретного користувача, сервер у відправить не тільки дані про нього, а також інформацію про те, як змінити його стан, наприклад, як змінити його ім’я або виконати видалення. Сенс Uniform Interface такий, що запити від різних клієнтів виглядають однаково, незалежно від того чи клієнт використовує chrome браузер, linux сервер, python скрипт, android застосунок чи щось інше.

— Client - server separation означає, що клієнт і сервер маються працювати незалежно один від одного. Комунікація проходить за допомогою запитів, ініційованих тільки клієнтом, та відповідями, які сервер посилає лише як

реакцію до запиту. Сервер очікує запити від клієнта. Сервер не віддає інформацію про ресурси сам по собі.

— Stateless означає, що сервер не запам’ятовує нічого про користувача API. Наприклад, він не має пам’ятати чи клієнт відправляв GET запит в минулому і які по яким саме ресурсам був запит. Кожен індивідуальний запит містить у собі всю необхідну інформацію серверу, для обробки і відправки відповіді, незалежно від інших запитів зроблених користувачем.

— Layered system означає, що між користувачем який посилає запит на представлення деякого ресурсу і сервером, який у відповідь відправляє дані, може бути деяка кількість серверів посередників. Вони можуть відповідати за безпеку системи, кешування або виконувати інші функції. Посередники не впливають не міняють початковий запит або відповідь. Клієнт не залежить від цих додаткових рівнів, якщо вони є, між клієнтом і фактичним сервером, який відповіжає на запит.

— Cacheable означає, що дані які сервер містять інформацію про їх кешування. Якщо так, то вона може містити номер версії. Це те, що робить кешування можливим: так як клієнт знає яку версію даних він має, що дозволяє пропустити етап завантаження вже закешованих даних. Користувач також повинен дізнаватися що дані застаріли, у такому разі буде надіслано запит на найбільш актуальне представлення даних з серверу.

— Code-on-demand не є обов’язковою умовою для API щоб бути RESTful. Клієнт може відправити запит на код для отримання відповіді з серверу у вигляді скрипту, для його виконання на стороні користувача.

4. ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

В цьому розділі проводиться опис структури і архітектури програмного забезпечення.

4.1 Опис функціональних можливостей застосунку

Перед тим як почати розробку системи, потрібно визначити які функціональні можливості вона має реалізовувати, які вхідні і вихідні дані мають бути, тощо. Для наглядної демонстрації функціональних можливостей системи була створена діаграма прецедентів системи. (Рисунок 4.1.1)

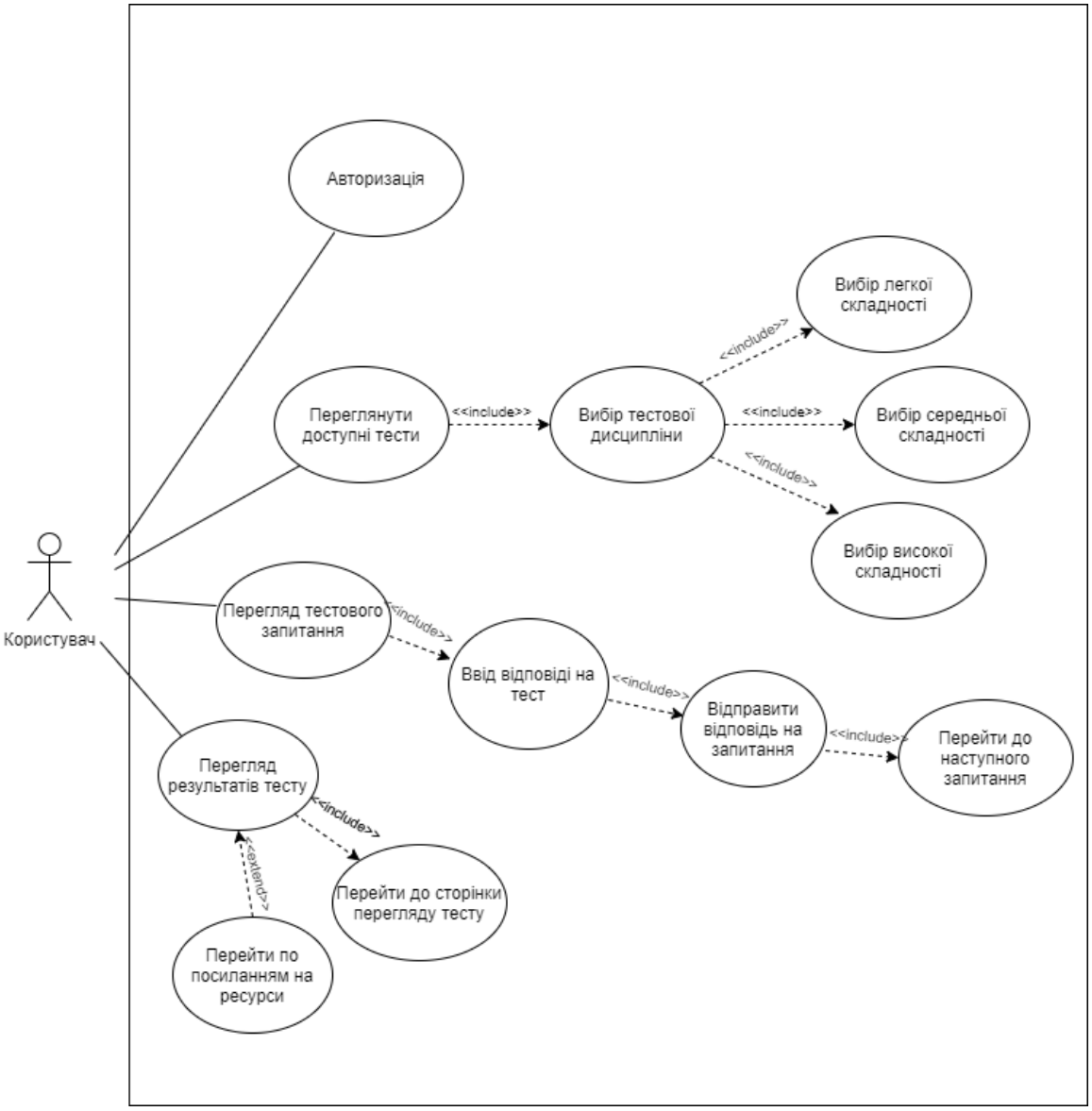

Рисунок 4.1.1 — Діаграма прецедентів системи

Як видно з діаграми, застосунок можна розділити на чотири типи екранів: сторінки авторизації, вибору тесту, питання і перегляду результатів.

Сторінка авторизації може представляти з себе екран з двома полями (ім’я, група) і кнопкою відправки даних. Після вводу необхідних даних відбувається перехід на сторінку вибору тесту.

Сторінка вибору з тесту це список тестів у вигляді карточок по доступним дисциплінам. Інформація при них приходить від сервера, у відповідь на відповідний запит. Кожен тест має 3 доступних рівні складності. Вона впливає на типи питань і самі питання, які зустрічаються під час тестування. Після вибору користувача посилається запит на генерацію тесту. Якщо операція успішна, сервер повертає інформацію про ідентифікатор згенерованого тесту, який потім використовується для запитів на тестові питання, і клієнт переходить на відповідний екран з ініціацією запиту на перше питання. У разі невдачі, сервер повідомляє про це застосунок, і користувачу виводиться відповідне повідомлення про це.

Інтерфейс сторінки тесту може відрізнятися в залежності від типу тесту, але деякі атрибути можуть залишатися незмінними. Серед них панель статусу тесту, яка представляє з себе деяку кількість кіл, вишикуваних в один ряд. Якщо коло зафарбоване зеленим, на питання було дано правильну відповідь, якщо червоним, то неправильну. Іншим елементом є текст запитання, який розташований трохи нижче. У нижній частині сторінки розташована кнопка, яка містить у собі декілька функції: перевірка поточного питання, перехід на наступне питання, перехід на сторінку перегляду результату.

Зараз система підтримує декілька типів питань. Кількість типів питань конкретного типу в тестуванні залежить від складності тесту. Нижче наведено список підтримуваних типів питань:

— Choose One – питання з вибором однієї правильної відповіді.

— Choose Several – питання з вибором декількох правильних відповідей.

— Match Items – співставлення термінів з твердженнями.

— Write Answer – запис правильної відповіді в поле.

— Fill Gaps – заповнення пропусків.

— Remove Wrong – питання з вибором всіх тверджень, які є хибними.

Коли від клієнтського застосунку посилається запит на наступне тестове завдання, сервер у відповідь додає інформацію про його тип для правильного відображення в інтерфейсі програми.

4.2 Опис структури проекту

Вихідний код програми складається з окремих структурних елементів і знаходиться в директорії lib (Рисунок 4.2.1)

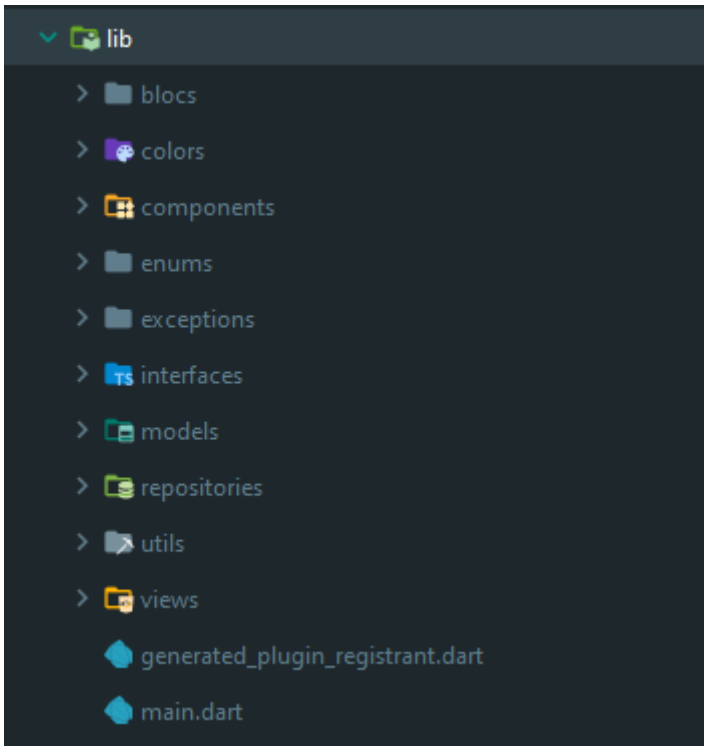

Рисунок 4.2.1 — Структура файлів проекту

Кожен модуль програми розміщений в своїй директорії, відповідно до його функціонального призначення:

- blocs – модулі побудовані відповідно до архітектури BLoC, що відповідають за просте і передбачуване управління станом окремих частин програми.
- colors – перелік змінних-кольорів для їх використання в користувацькому інтерфейсі.
- components – логічно виділені частини UI, з яких будується інтерфейс, через який проходить взаємодія з системою.

- enums – перераховуванні типи, спеціальний тип класу, який використовується для представлення деякої кількості константних значень. Вони мають деякі обмеження в мові Dart, такі як неможливість наслідування і явного створення екземплярів.
- exceptions – описи класів виключних ситуацій, які використовуються в застосунку.
- interfaces – описи інтерфейсів, які використовуються для реалізації dependency injection.
- models – класи сутностей, якими оперує застосунок. Також існує спосіб створення об’єктів за допомогою JSON.
- repositories – відповідають за ініціацію запитів до серверу. Містять всі необхідні методи для будь-яких запитів.
- utils – функції-утиліти
- views – віджети які знаходяться в корені дерева кожної з сторінок. Тут проходить основна взаємодія з bloc і описується користувацький інтерфейс.
- generated_plugin_registrant – згенерований файл для роботи застосунку в браузері.
- main – вхідна точка як для Dart, так і для Flutter застосунків.

способом. Крім цього, навігація по сторінкам тестового питання і результатів тесту виконується за допомогою Navigator API.

Логіка комунікації застосунку з сервером знаходиться в класах репозиторіях. Реалізовані дані класи за допомогою пакету http. Для всіх необхідних запитів створені відповідні методи. Однією з особливостей є використання async/await, які дозволяють дуже зручно обробляти асинхронні операції. Асинхронні операції дозволяють програмі виконувати роботу в час очікування на завершення іншої операції. Найбільш поширеними асинхронними операціями є:

— Отримання даних з мережі

— Записування даних в БД

— Зчитування даних з файлу

Для виконання цих операцій в мові програмування Dart використовується класс Future або ключові слова async та await. Блоки синхронних операцій блокують інші поки вони повністю не виконаються. Функції, які виконують тільки синхронні операції називаються синхронними. Натомість, асинхронні операції дозволяють виконуватись іншим операціям до їх завершення. Функції, які виконують хоча б одну таку операцію, називаються асинхронними. Розглянемо способи виконання даних операцій.

Одним з базових API мови програмування Dart для асинхронного програмування є об’єкти Future. В загальному, вони дуже схожі на аналоги з інших мов програмування. Так як в мові існує можливість використання async/await, є можливість відмовитись від явного використання Future. Натомість, зустріч з даними об’єктами неминуча.

Для кращого розуміння можна думати про ці об’єкти, як про коробки з даними, які початково закриті. Через деякий час коробка відкривається і всередині може бути як дані, так і об’єкт помилки. Об’єкт може бути в одному з трьох станів (Рисунок 4.3.1):

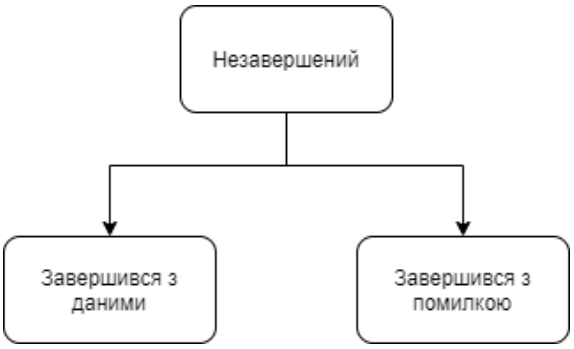

Рисунок 4.3.1 — Можливі стани об’єкту Future

— Незавершений. Коробка закрита.

— Завершений зі значенням. Коробка відкрита і дані готові.

— Завершений з помилкою. Коробка відкрита, але щось пішло не так.

Більшість коду, який працює з Future, написаний для обробки наведених трьох станів. Ви отримуєте об’єкт, і ви повинні описати що потрібно робити до того, як «коробка» відкриється, що робити коли приходить значення, що робити при отриманні помилки. Цей патерн зустрічається дуже часто. В мові програмування Dart є так званий Event Loop для роботи з асинхронними операціями. Future це просто

вбудований інструментарій для роботи з ним. Код застосунку на Dart виконується в єдиному потоці. Коли застосунок працює, цей потік крім виконання синхронного коду, бере події з Event Loop і оброблює їх. Припустимо, що в застосунку є кнопка завантаження. Коли користувач натискає на неї, починається завантаження зображення. Спочатку з'являється події натиску на кнопку. Коли Event Loop отримує цю подію, викликається її обробник. Обробник використовує http бібліотека для виконання запиту, який повертає об'єкт Future. Після його отримання, очікується результат. В момент очікування можуть відбуватися інші події, користувач може працювати з інтерфейсом, Event Loop продовжує роботу. Нарешті, дані зображення отримані, і Future завершується зі значенням, викликаючи функцію зворотнього виклику.

Серверна частина застосунку відповідає за обробку запитів клієнтів, надає усі необхідні дані для роботи застосунку і записує результати роботи системи в базу даних для подальшого їх перегляду. Вона побудована на реверс проксі сервері Nginx, який відповідає за routing запиті – перенаправлення на необхідний застосунок. Потрібна для роботи розробленого в роботі застосунку програма реалізує генерацію тестів з використання понятійно-тезисної моделі, яка обговорювалася в попередніх розділах. Саме ця частина системи має прямий доступ до бази даних і навчальних матеріалів, на основі яких будуються тести.

Логіка програмного застосунку за допомогою застосування патернів BLoC і Provider. BLoC клас реалізований для кожної сторінки і типу питання, які відповідають за обробку дій користувача і комунікації з сервером. В ході проектування застосунку були виділені основні сутності, які використовуються в проекті (4.2.1).

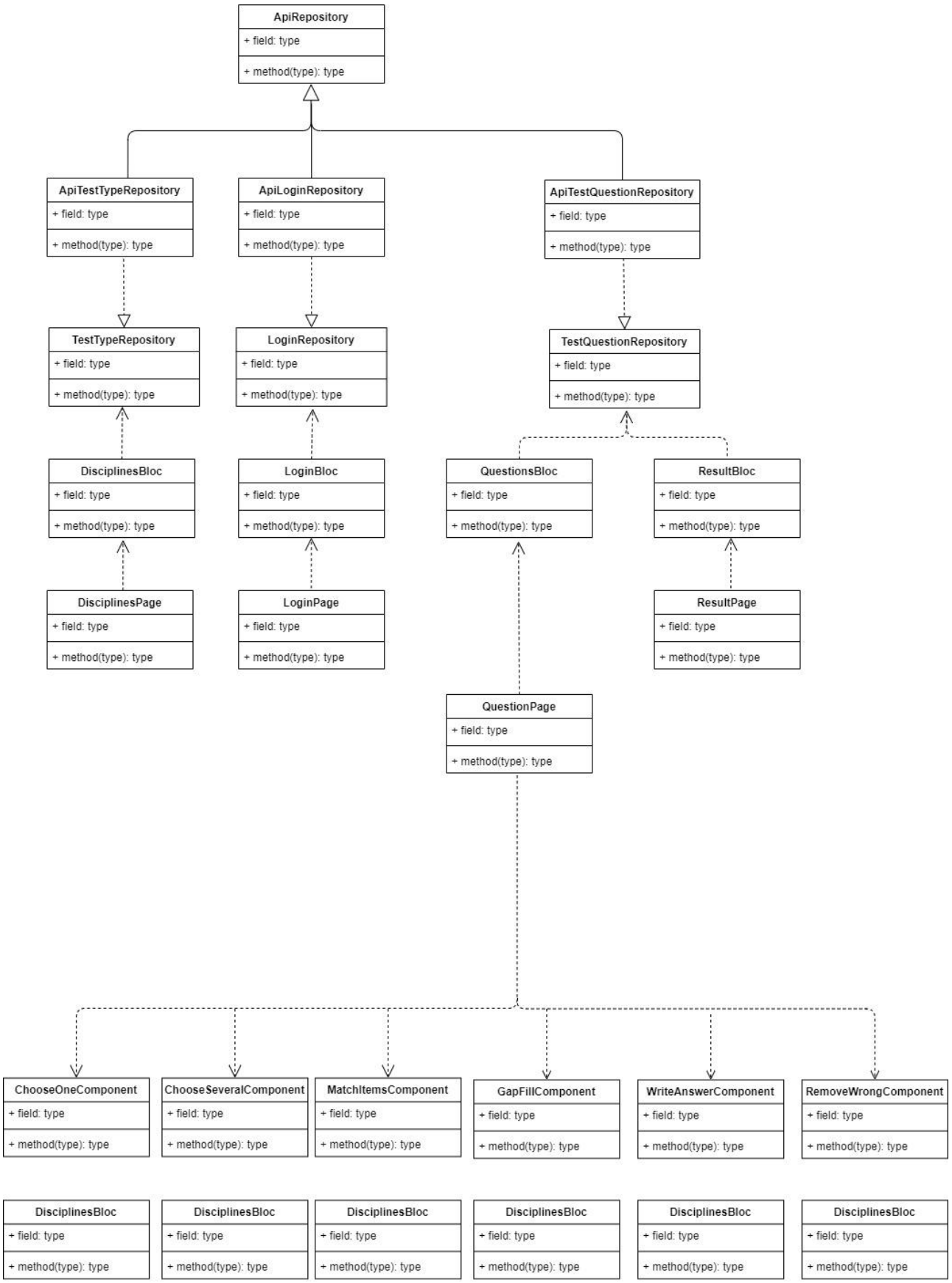

Рисунок 4.2.1 – Спрощена діаграма класів проекту

Для створення єдиної кодової бази для мобільних пристроїв і веб-браузерів довелося враховувати декілька особливостей кожної з плаформ. Однією з таких особливостей є відправка запитів до серверу, так як це відбувається по різному на перелічених платформах. На мобільних платформах дані cookie потрібно посилати вручну, тоді як в браузері ручне задання цього параметру заблоковано в цілях безпеки.

Для правильного врахування цих нюансів потрібно в момент роботи застосунку визначати платформу, на якій в даний момент часу виконується код. Під озвучену задачу підходить глобальна константа kIsWeb, яку надає середовище виконання фреймворку Flutter. Ця константа має значення істини, якщо застосунок був скомпільовано для використання у вебі. Така реалізація використовує той факт, що в мові програмування JavaScript не підтримуються цілочисельні типи. В таких середовищах, дробові і цілі числа приводяться до одного об’єкту. Тому дробове число 0.0 виходить ідентичним до цілого числа 0.

Також відмінною рисою браузерів є їхні механізми безпеки, які відсутні при розробці на мобільні платформи. Однією з проблем під час реалізації функціоналу був так званий Cross-Origin Resource Sharing, що потребувало деяких змін зі сторони серверу під час розробки клієнтського застосунку.

5. РОБОТА КОРИСТУВАЧА З ПРОГРАМНОЮ СИСТЕМОЮ

5.1 Системні вимоги

Для роботи застосунку достатньо смартфону з операційною системою Android або IOS. Також система працює в новітніх усіх веб-браузерів, з дозволеною роботою мови програмування JavaScript, з підтримкою Canvas API.

5.2 Сценарій роботи користувача з застосунком і опис інтерфейсу

У даному розділі будуть розглянуті основні етапи роботи користувача з програмною системою.

При запуску застосунку користувача зустрічає сторінка входу в систему. Для продовження роботи потрібно ввести своє ім’я і номер групи. (Рисунок 5.2.1)

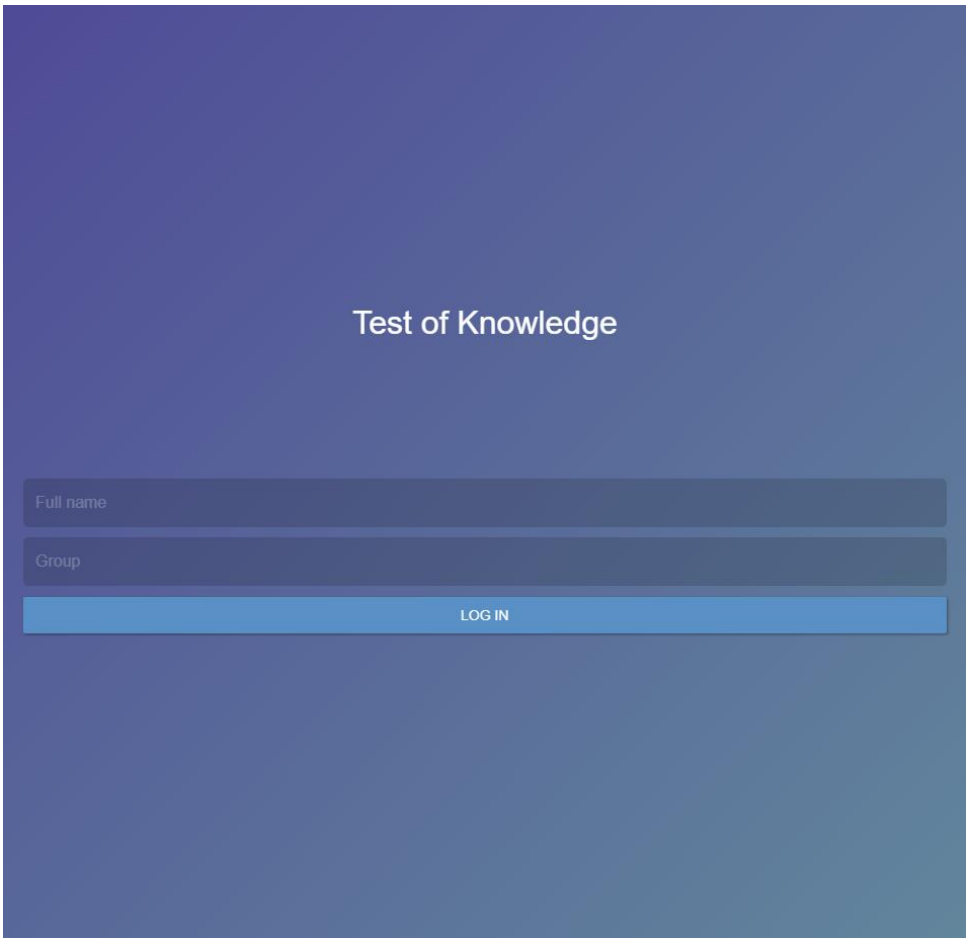

Рисунок 5.2.1 – Початкова сторінка веб-додатку

Після вводу необхідних даних, натиснення відповідної кнопки “LOG IN” і успішної обробки запиту відбувається перехід на сторінку вибору тесту. Інтерфейс представляє з себе список дисциплін, по яким можна пройти тестування. Елементи списку представляють окрему дисципліну. Кожна дисципліна дозволяє вибирати між трьома рівнями складності. Для кожного предмету існує три кнопки: «Easy» - легкий, «Medium» - середній і «Hard» - складний. Коли користувач натискає на будь-яку з кнопок, відправляється запит на генерацію бажаного тесту, як що він успішний, то виконується перехід на сторінку першого тестового питання. Інтерфейс цієї сторінки продемонстровано на Рисунку 5.2.2.

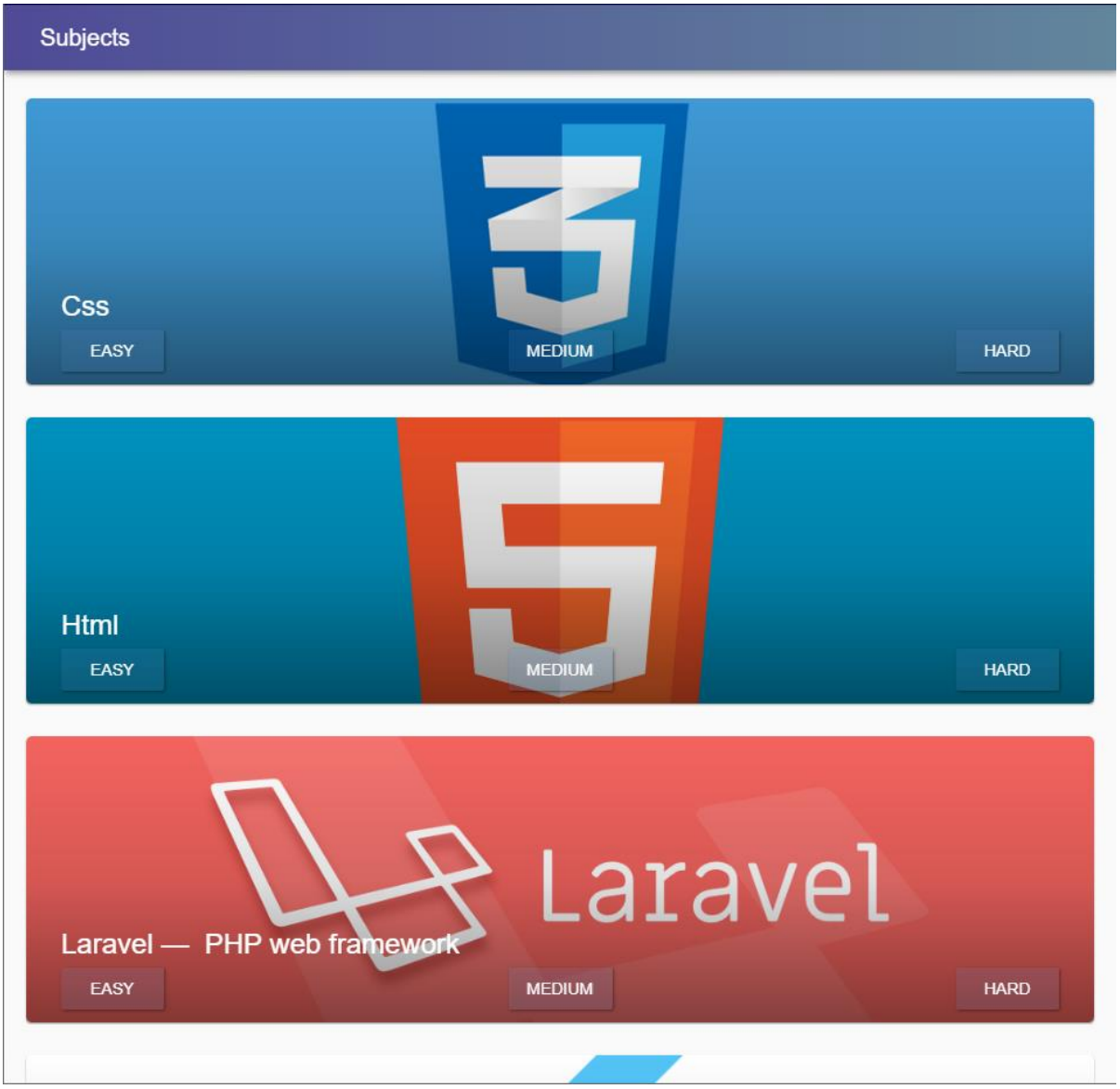

Рисунок 5.2.2 – Сторінка вибору тесту.

На сторінках тестових питань користувач може переглянути власне текст питання, варіанти відповідей (якщо вони передбачуються типом питання). У верхній частині

сторінки розташований індикатор прогресу тестування, який містить інформіцію про загальну кількість питань (кількість кругів) і статус кожного з них. Салатовий колір меж круга означає, що користувач працює саме з цим питанням. Якщо круг зафарбований зеленим кольором, це означає що відповідь на питання була дана вірно, а якщо червоним – невірно. Внизу сторінки знаходиться кнопка для перевірки і переходу на наступні етапи тестування. (Рисунок 5.2.3)

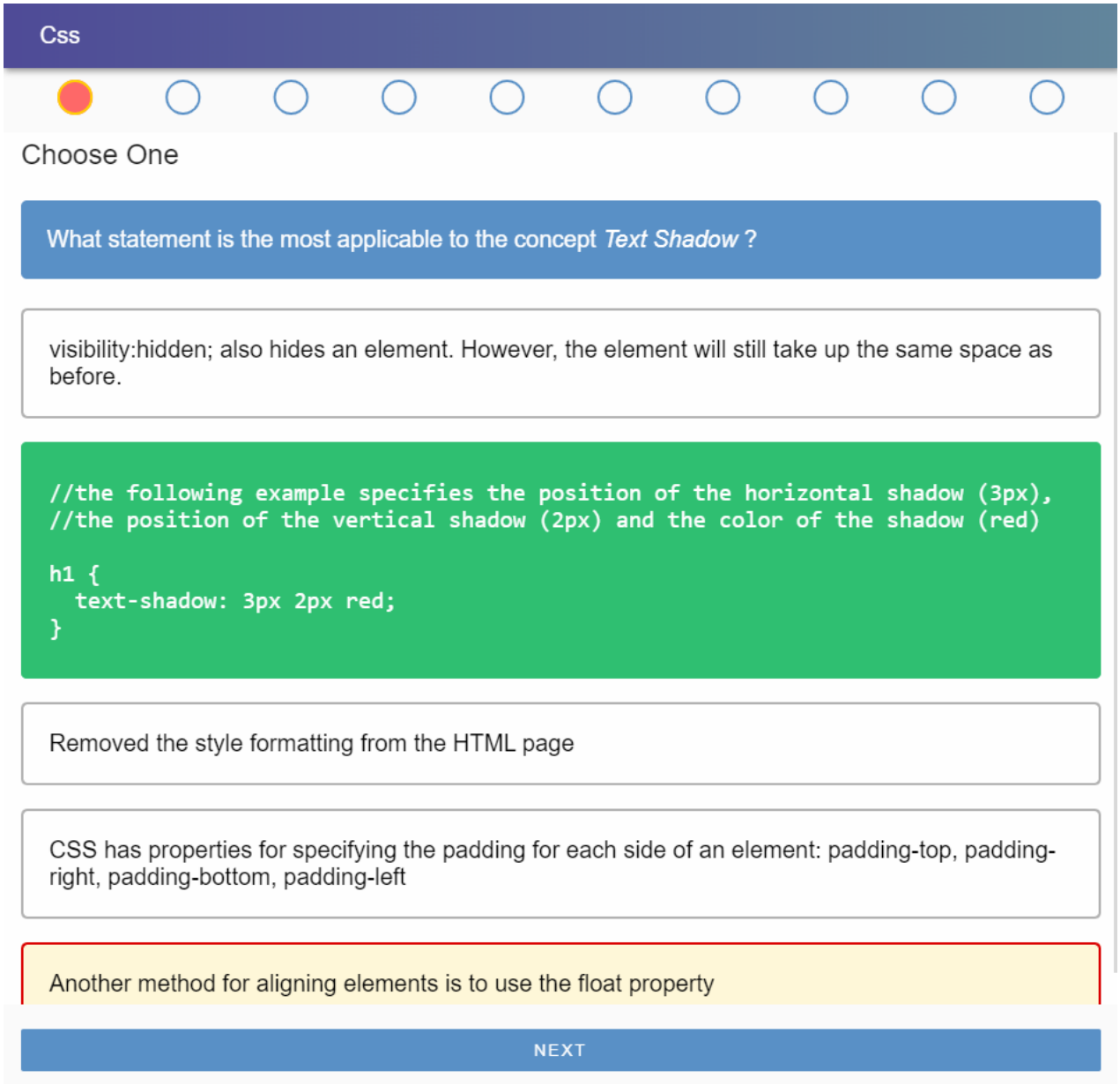

Рисунок 5.2.3 – Сторінка тестового питання.

Як видно з рисунку вище, зона вводу відповіді на питання знаходиться в центральній частині сторінки. Інтерфейс тестового завдання може різнитися, в залежності від типу запитання. Розглянемо інтерфейси різних видів запитань:

— Choose One, вибір однієї правильної відповіді (Рисунок 5.2.4)

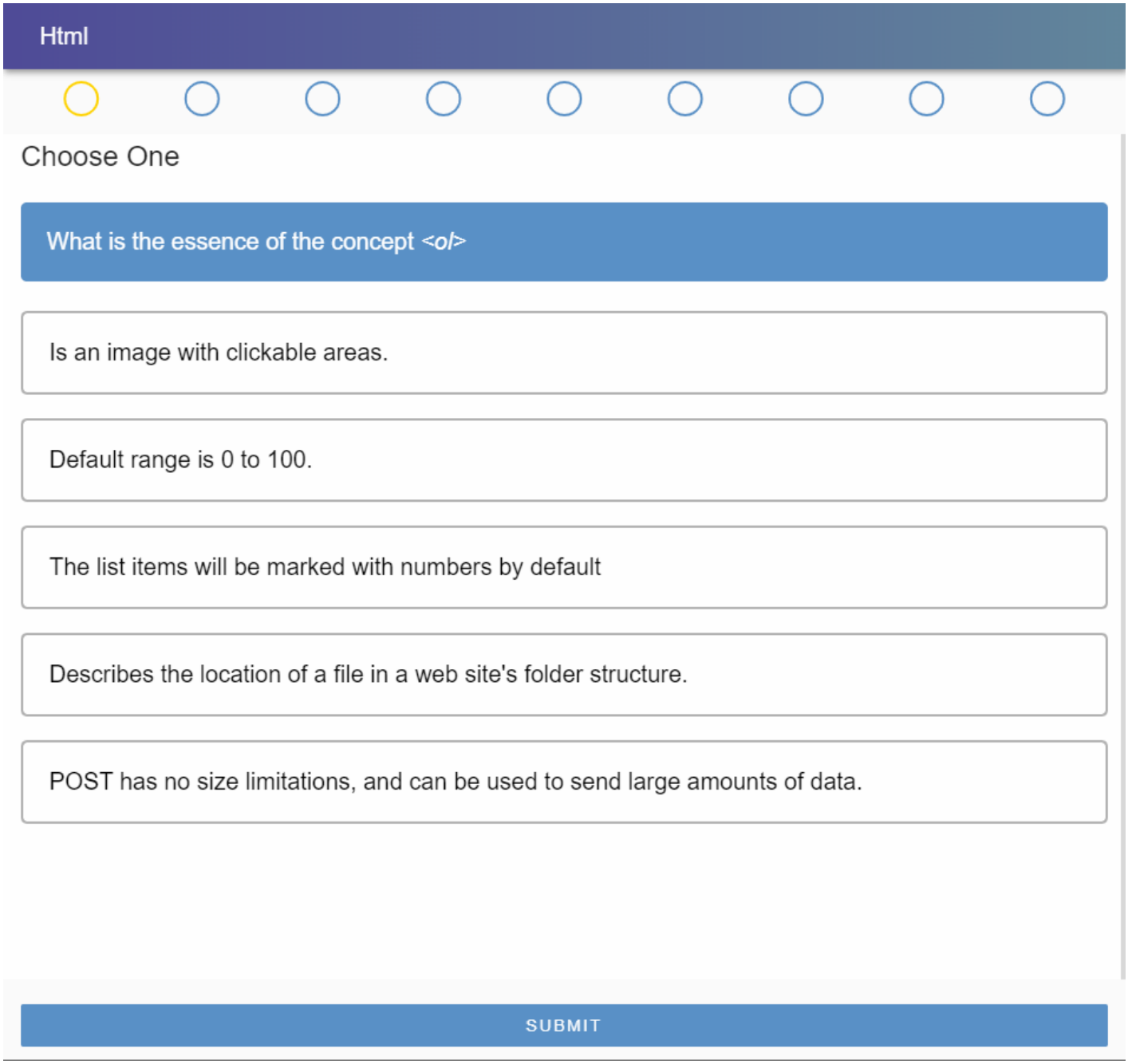

Рисунок 5.2.4 – Інтерфейс тестового питання виду Choose One.

— Choose Several, вибір декількох правильних відповідей (Рисунок 5.2.5)

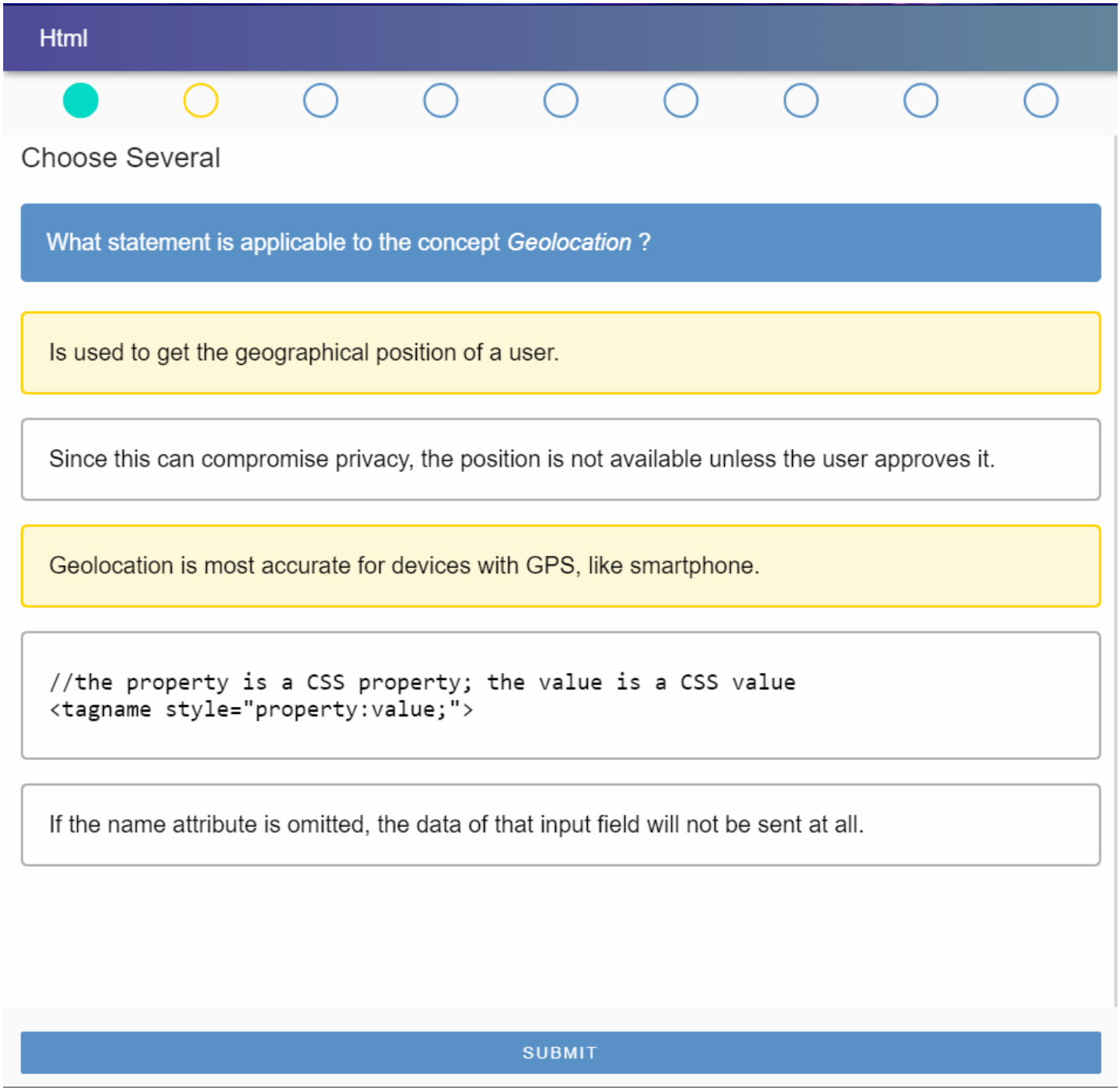

Рисунок 5.2.5 – Інтерфейс тестового питання виду Choose Several.

— Write Answer, написання правильної відповіді (Рисунок 5.2.6)

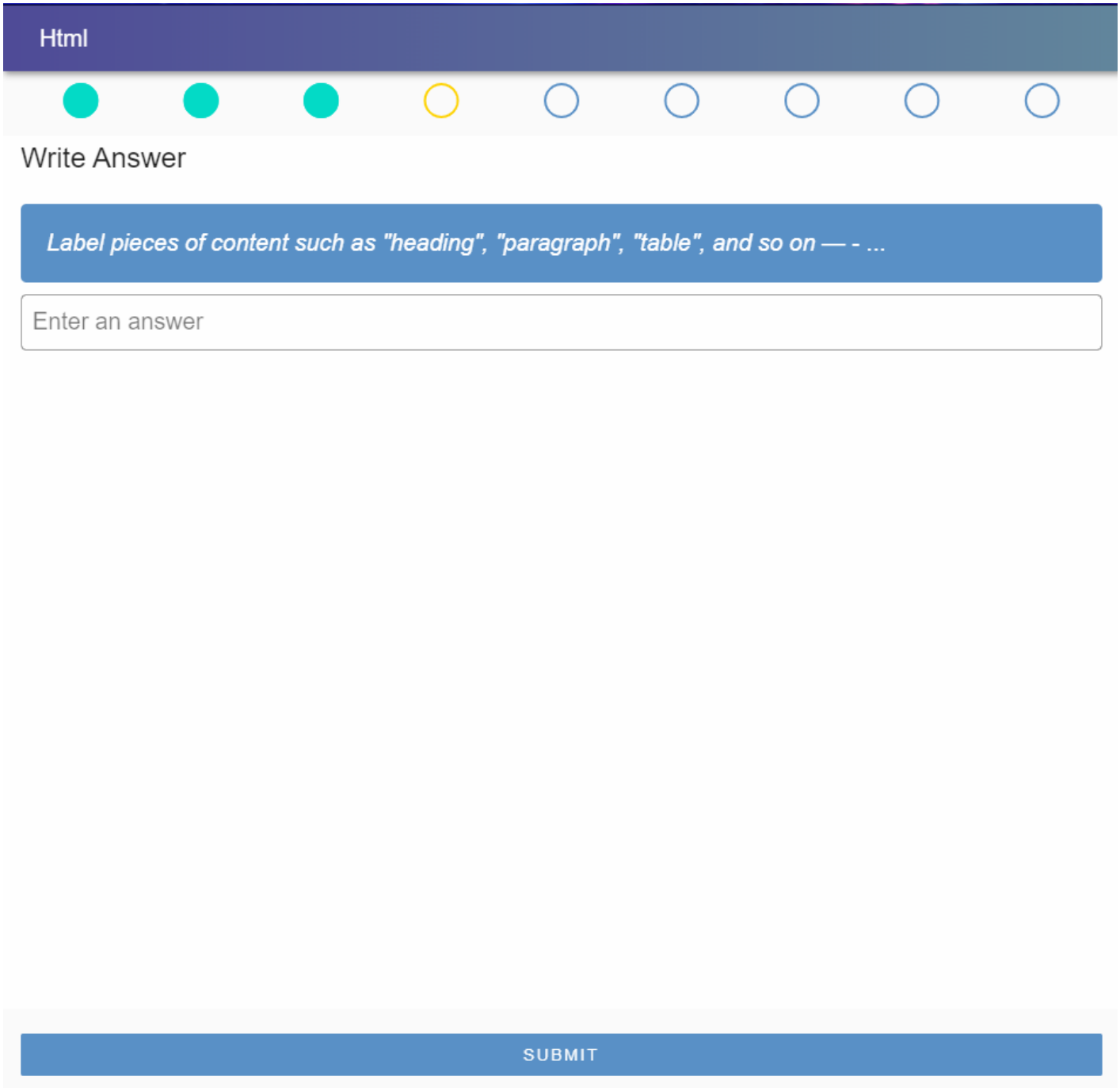

Рисунок 5.2.6 – Інтерфейс тестового питання виду Write Answer.

— Match Items, співставлення термінів і тверджень (Рисунок 5.2.7)

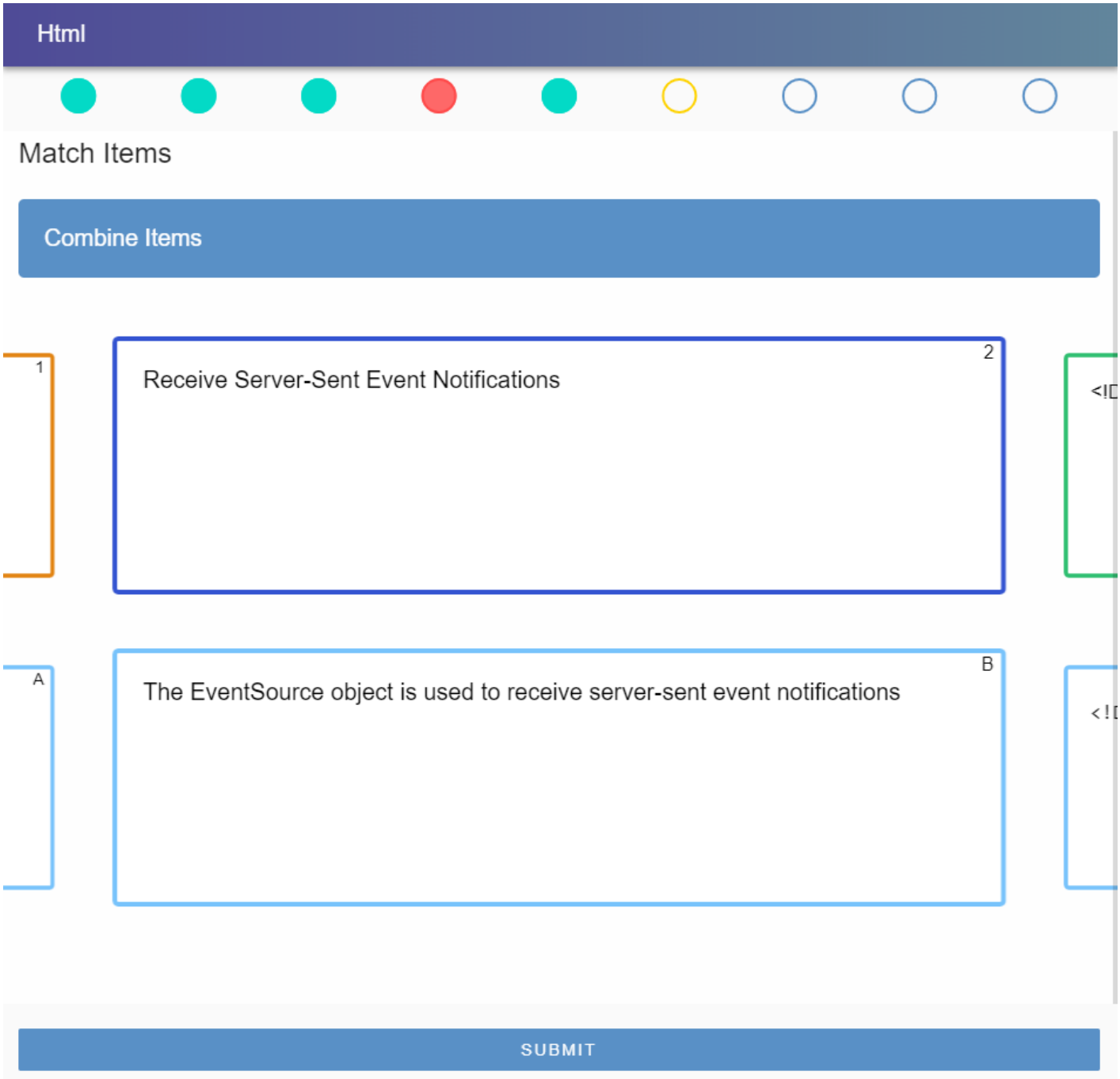

Рисунок 5.2.7 – Інтерфейс тестового питання виду Match Items.

— Fill Gaps, співставлення термінів і тверджень (Рисунок 5.2.8)

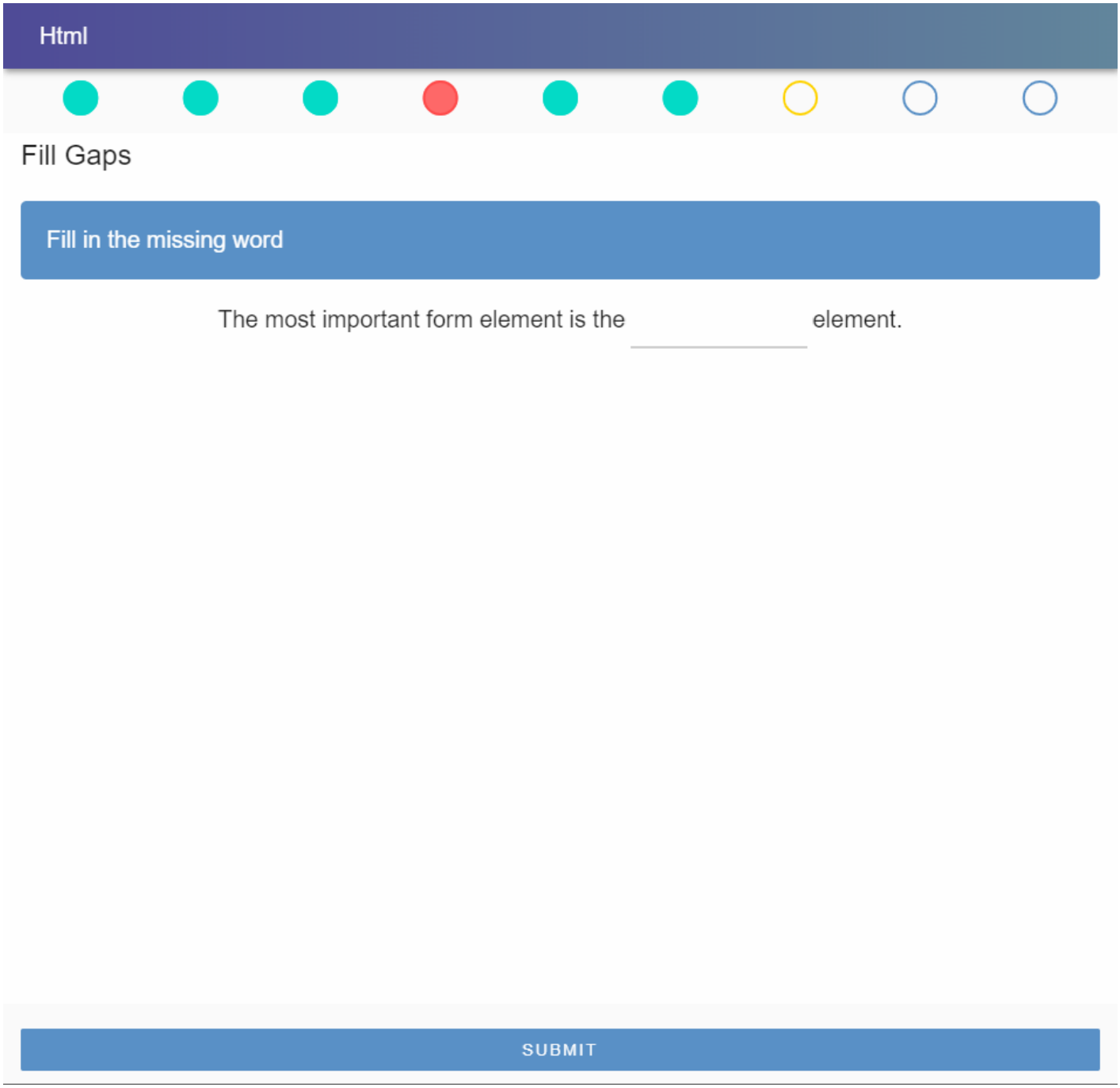

Рисунок 5.2.8 – Інтерфейс тестового питання виду Match Items.

Після проходження тесту, користувачеві виводиться інформація про результат тестування. (Рисунок 5.2.9)

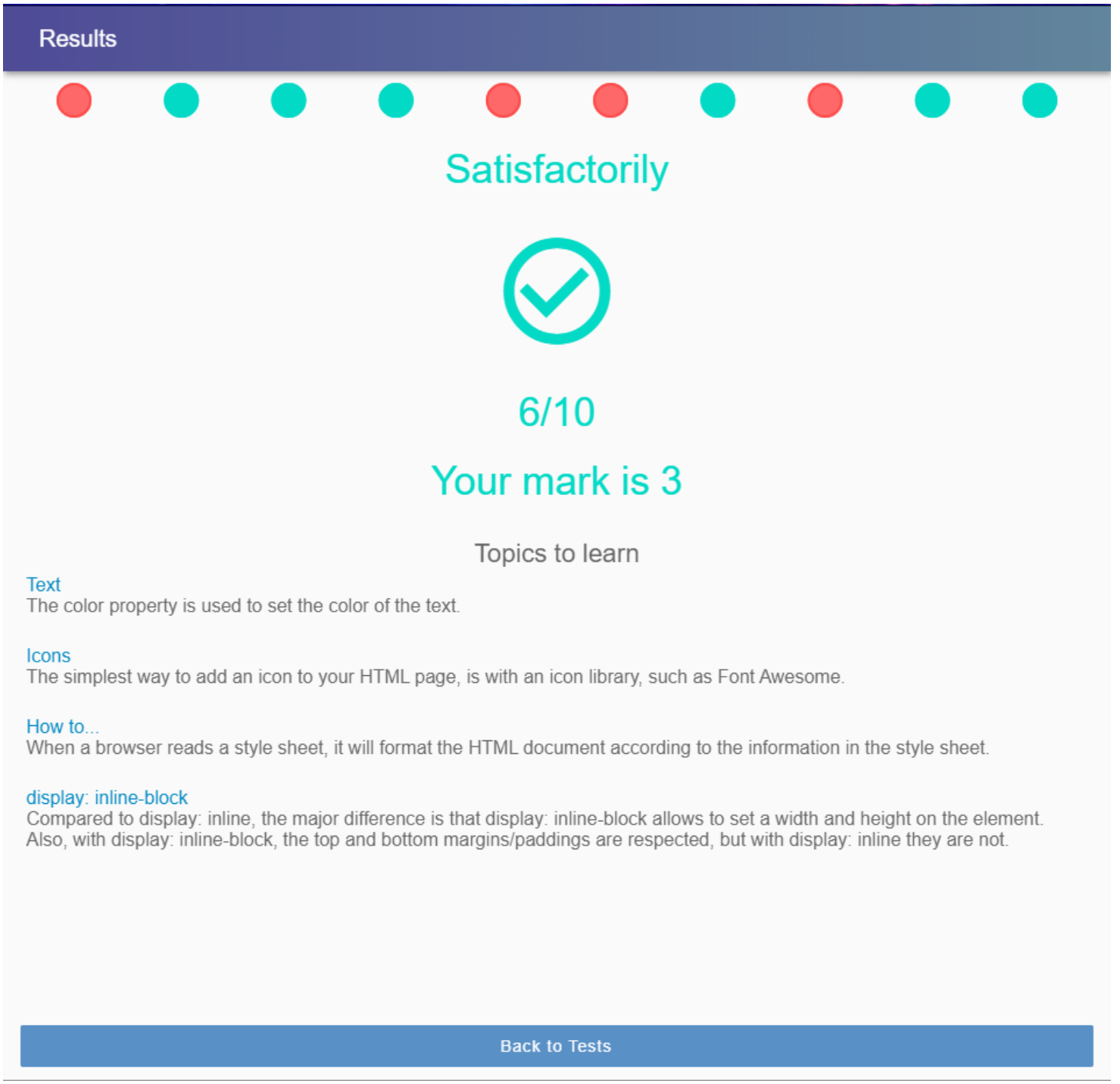

Рисунок 5.2.9 – Сторінка результату тестування.

Як можна побачити з рисунку, у верхній частині сторінки можна побачити вже описаний індикатор прогресу, трохи нижче інформація про результат тестування: частка правильних відповідей і кількість набраних балів. Далі йде список посилань на освітні ресурси по темам, знання по яким варто покращити. У разі бездоганного проходження тесту список відсутній. У нижній частині сторінки знаходиться кнопка, яка відповідає за повернення на сторінку вибору тесту для повторного тестування.

ВИСНОВКИ

У ході виконання роботи були досліджені література на тему роботи, предметна область, документація програмних засобів, які були використані для розробки алгоритму. Були проаналізовані системи аналоги, цільова аудиторія створюваного програмного продукту. В результаті дослідження було визначено, що аналогічні системи не мають захисту від недобросовісного тестування, так як вони використовують статичні завдання для перевірки знань по темі.

Після аналізу стану предметної області було вирішено організувати можливість роботи застосунку через інтернет, а також реалізувати застосунок для декількох платформ для більшої доступності продукту.

Була поставлена задача розробка програмного продукту. Був обґрунтований вибір технологій для створення програмної системи. Вибір був за платформою Flutter, що забезпечує швидку і якісну розробку застосунку, а також збільшує гнучкість і простоту підтримки системи. Для побудови архітектури проекту був використаний патерн проектування BLoC. Застосунок було вирішено базувати на основі існуючого REST API для проходження тестів, які генеруються за допомогою ПТМ. Такий підхід значно зменшує затрати часу для створення тестових завдань і ймовірність недобросовісного проходження тесту. Був спроектований і реалізований алгоритм роботи системи.

В результаті випробувань системи було визначено, що система працює правильно і повністю реалізовує функціонал, описаний в поставленій задачі. Це дозволяє користувачам отримувати правильні результати оцінювання. Система дозволяє користувачеві проходити контроль знань по дисциплінам онлайн, в умовах дистанційної освіти, без необхідності фізичної присутності в навчальному закладі. Система пропонує пройти оцінювання знань за будь-якою з доступних дисциплін, з вибором однієї з трьох складностей.

Цільовою аудиторією програмної системи студенти на які знаходяться як на денній так і заочній формі навчання.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Офіційна документація Flutter [Електронний ресурс] – Режим доступу до ресурсу: https://flutter.dev/docs.

2. Grahame Moore M. Handbook of Distance Education / M. Grahame Moore, W. G. Anderson., 2003.

3. Елизаренко Г.Н. Проектирование компьютерных курсов обучения: концепция, язык, структура. НТУУ «КПИ». Киев, 2001.

4. Slavomir Stankov, Branko Žitko and Ani Grubišić. Ontology as a Foundation for Knowledge Evaluation in Intelligent E-learning Systems. AIED’05 Workshop SW-EL’05: Applications of Semantic Web Technologies for E-Learning. Papers of 12th International Conference on Artificial Intelligence in Education (AIED 2005). Amsterdam, 2005. http://hcs.science.uva.nl/AIED2005/W3proc.pdf

5. Berners-Lee T. "Spinning the Semantic Web: Bringing the World Wide Web to Its Full Potential", The MI Press, 2005

6. Артур К. Как изучать Библию. Санкт-Петербург, 1998.

7. Gordon D. Fee, Douglas Stuart. How to Read the Bible For All its Worth: A Guide to Understanding the Bible. Micchigan, 1982.

8. Gagarin A., Tytenko S. Complex model of educational hypermedia environment for ongoing learning // Образование и виртуальность – 2007. Сборник научных трудов 11-й Международной конференции Украинской ассоциации дистанционного образования / Под общ. ред. В.А. Гребенюка, Др Киншука и В.В. Семенца.– Харьков-Ялта: УАДО, 2007.– С. 140-145

9. Гагарін О.О., Титенко С.В. Проблеми створення гіпертекстового навчаючого середовища // Вісник Східноукраїнського національного університету імені Володимира Даля №4 (110) 2007 Ч.2 - Луганськ 2007 - С. 6-15.

10. Лабораторія СЕТ – Віртуальна лабораторія новітніх інформаційних технологій. Дослідження в області дистанційного навчання http://www.setlab.net

11. Офіційна документація мови програмування Dart [Електронний ресурс] – Режим доступу до ресурсу: https://dart.dev/guides.

12. Документація до пакету flutter_bloc [Електронний ресурс] – Режим доступу до ресурсу: https://bloclibrary.dev/.
13. Orozco J. V. RESTful Services: what they are, and what they aren't [Електронний ресурс] / Juan Villalobos Orozco. – 2019. – Режим доступу до ресурсу: https://www.brainstobytes.com/restful-services-what-they-are-and-what-they-are-not/.

ДОДАТОК А

Інструментальні засоби для тестування знань по різним дисциплінам з програмною генерацією питань

Специфікація

УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ТІ62628_20Б 1

Аркушів 1

Київ 2020

Позначення	Найменування	Примітки
Документація		
УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ТІ6 2628_20Б	Записка.docx	Текстова частина дипломної роботи
Компоненти		
УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ТІ6 2628_20Б 2	main.dart, login_bloc.dart disciplines_bloc.dart login_bloc.dart question_bloc.dart result_bloc.dart	Основні компоненти

ДОДАТОК Б

Інструментальні засоби для тестування знань по різним дисциплінам з програмною генерацією питань

Текст програми

УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ТІ62628_20Б 2

Аркушів 5

2020

```
import 'package:flutter/material.dart';
import 'package:flutter/rendering.dart';

import 'package:flutter_bloc/flutter_bloc.dart';
import 'package:semantic_api_consumer/blocs/index.dart';
import 'package:semantic_api_consumer/repositories/index.dart';

import 'package:semantic_api_consumer/views/index.dart';

// Enter point of the program
void main() {
  debugPaintSizeEnabled = false;
  BlocSupervisor.delegate = SimpleBlocDelegate();
  runApp(SemanticPortalApiConsumerApp());
}

class SemanticPortalApiConsumerApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      initialRoute: '/',
      routes: {
        '/': (context) => BlocProvider(
          create: (context) => LoginBloc(
              repository: ApiTestTypeRepository()
          ),
          child: LoginPage(),
        ),
        'subjects': (context) => BlocProvider(
          create: (context) => DisciplinesBloc(
            repository: ApiTestTypeRepository()
          ),
          child: SubjectPage(),
        )
      },
      debugShowCheckedModeBanner: false,
      title: 'Test of Knowledge',
    );
  }
}

import 'dart:async';
import 'package:http/http.dart';
import 'package:meta/meta.dart';
import 'package:bloc/bloc.dart';
import 'package:semantic_api_consumer/exceptions/index.dart';
import 'package:semantic_api_consumer/interfaces/index.dart';
import 'package:semantic_api_consumer/models/index.dart';
import 'package:semantic_api_consumer/blocs/disciplines/bloc.dart';

class DisciplinesBloc extends Bloc<DisciplinesEvent, DisciplinesState> {
  @override
```

```
  DisciplinesState get initialState => DisciplinesInitialState();
  final TestTypeRepository _testTypeRepository;
  DisciplinesBloc({@required TestTypeRepository repository}):
_testTypeRepository = repository;

  List<TestTypeCardData> _cached;
  @override
  Stream<DisciplinesState> mapEventToState(DisciplinesEvent event) async* {
    if (event is DisciplinesFetchData) {
      yield* _mapToFetchDisciplines();
    } else if (event is DisciplinesRequestTestGenerate) {
      yield* _mapToGenerateDisciplines(event);
    }
  }

  Stream<DisciplinesState> _mapToFetchDisciplines() async* {
    yield DisciplinesPending();
    try {
      List<TestTypeCardData> response = _cached ?? await
_testTypeRepository.fetch();
      _cached = response;
      Disciplines disciplines = Disciplines(data: response);
      yield DisciplinesLoadingSucceed(disciplines);
    } on FetchDataFailedException catch (e) {
      yield DisciplinesLoadingError(e.message);
    } on ClientException catch (e) {
      print('Http client error! ${e.toString()}');
      yield DisciplinesLoadingError(e.toString());
    }
  }

  Stream<DisciplinesState>
_mapToGenerateDisciplines(DisciplinesRequestTestGenerate event) async* {
    yield DisciplinesPending();
    try {
      GenerateTestResponseData responseData = await
_testTypeRepository.setupTest(event.disciplineName, event.difficulty);
      yield DisciplinesGenerateTestSucceed(responseData);
    } on FetchDataFailedException catch (e) {
      yield DisciplinesGenerateTestFailed(e.message);
    } on ClientException catch (e) {
      print('Http client error! ${e.toString()}');
      yield DisciplinesLoadingError(e.toString());
    }
  }

}

import 'dart:async';
import 'package:http/http.dart';
import 'package:meta/meta.dart';
import 'package:bloc/bloc.dart';
```

```
import 'package:semantic_api_consumer/exceptions/index.dart';
import 'package:semantic_api_consumer/interfaces/index.dart';
import 'package:semantic_api_consumer/models/index.dart';
import 'package:semantic_api_consumer/blocs/login/bloc.dart';

class LoginBloc extends Bloc<LoginEvent, LoginState> {
  @override
  LoginState get initialState => LoginInitialState();
  final LoginRepository _loginRepository;
  LoginBloc({@required LoginRepository repository}): _loginRepository =
repository;

  @override
  Stream<LoginState> mapEventToState(LoginEvent event) async* {
    if (event is LoginRequestAction) {
      yield LoginPending();
      try {
        bool isLoggedIn = await _loginRepository.login(event.name,
event.group);
        yield LoginSucceed(LoginData(succeed: isLoggedIn));

        if (isLoggedIn) {
          yield LoginSucceed(LoginData(succeed: isLoggedIn));
        } else {
          yield LoginError("Couldn't login, please try again");
        }

      } on FetchDataFailedException catch (e) {
        yield LoginError(e.message);
      } on ClientException catch (e) {
        print('Http client error! ${e.toString()}');
        yield LoginError(e.toString());
      }
    }
  }
}

import 'dart:async';
import 'package:http/http.dart';
import 'package:meta/meta.dart';
import 'package:bloc/bloc.dart';
import 'package:semantic_api_consumer/blocs/question/question_event.dart';
import 'package:semantic_api_consumer/blocs/question/question_state.dart';
import
'package:semantic_api_consumer/exceptions/fetch_data_failed/fetch_data_failed
.dart';
import
'package:semantic_api_consumer/models/test_question/test_question.dart';

class QuestionBloc extends Bloc<QuestionEvent, QuestionState> {
  @override
  QuestionState get initialState => QuestionInitialState();
```

```
  final String _uuid;
  final TestQuestionRepository _repository;
  TestQuestion _question;

  QuestionBloc({ @required TestQuestionRepository repository, @required
String uuid}): _repository = repository, _uuid = uuid;

  @override
  Stream<QuestionState> mapEventToState(QuestionEvent event) async* {
    if (event is QuestionLoadDataEvent) {
      yield* _mapToFetchQuestion();
    } else if (event is QuestionSubmitDataEvent) {
      yield* _mapToSubmitQuestion(event);
    }
  }

  Stream<QuestionState> _mapToFetchQuestion() async* {
    yield QuestionPending();
    try {
      TestQuestion question = await _repository.fetchTestQuestion(_uuid);
      _question = question;
      yield QuestionLoadDataSucceed(question);
    }  on FetchDataFailedException catch(e) {
      yield QuestionLoadDataFailed(e.message);
    } on ClientException catch (e) {
      print('Http client error! ${e.toString()}');
      yield QuestionLoadDataFailed(e.toString());
    }
  }

  Stream<QuestionState> _mapToSubmitQuestion(QuestionSubmitDataEvent event)
async* {
    yield QuestionPending();
    try {
      TestQuestion question = await _repository.checkTestQuestion(_uuid,
_question.type, event.json);
      _question = question;
      yield QuestionSubmitDataSucceed(question);
    } on FetchDataFailedException catch(e) {
      yield QuestionSubmitDataFailed(e.message);
    } on ClientException catch (e) {
      print('Http client error! ${e.toString()}');
      yield QuestionSubmitDataFailed(e.toString());
    }
  }
}

import 'dart:async';
import 'package:http/http.dart';
import 'package:meta/meta.dart';
import 'package:bloc/bloc.dart';
import 'package:semantic_api_consumer/blocs/index.dart';
```

```
import
'package:semantic_api_consumer/exceptions/fetch_data_failed/fetch_data_failed
.dart';
import 'package:semantic_api_consumer/models/index.dart';

class ResultBloc extends Bloc<ResultEvent, ResultState> {
  @override
  ResultState get initialState => ResultInitialState();
  final ResultRepository _resultRepository;
  ResultBloc({@required ResultRepository repository}): _resultRepository =
repository;

  @override
  Stream<ResultState> mapEventToState(ResultEvent event) async* {
    if (event is ResultRequestAction) {
      yield ResultPending();
      try {
        ResultData resultData = await
_resultRepository.getTestResult(event.uuid);
        yield ResultSucceed(resultData);
      } on FetchDataFailedException catch(e) {
        yield ResultFailed(e.message);
      } on ClientException catch (e) {
        print('Http client error! ${e.toString()}');
        yield ResultFailed(e.toString());
      }
    }
  }

}
```

ДОДАТОК В

Інструментальні засоби для тестування знань по різним дисциплінам з програмною генерацією питань

Опис програми

УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ТІ62628_20Б З

Аркушів 9

Київ 2020

АНОТАЦІЯ

Додаток містить опис клієнтської частини системи «Test of Knowlegde». Застосунок виконує задачі, поставленні в розділі 1, а саме:

— Можливість вводу імені і групи на стартовій сторінці застосунку.
— Відображення доступних категорій для тестування і можливість вибору дисципліни.
— Можливість вибору складності дисципліни.
— Реалізація сторінки тесту з вибором однієї правильної відповіді.
— Реалізація сторінки тесту з вибором декільком правильних відповідей.
— Реалізація сторінки тесту з співставленням термінів до тверджень.
— Реалізація індикатора статусу проходження тесту (кількість питань в тесті, стан кожного питання окремо)
— Відображення результату тестування.
— Відображення списку посилань на інформаційні ресурси по темам, по яким були допущені помилки.

Застосунок реалізовано з використанням платформи Flutter.

ЗМІСТ

ЗАГАЛЬНІ ВІДОМОСТІ

В додатку міститься відомості про застосунок, який був розроблений у ході виконання праці, що виконує поставлену задачу цієї роботи.

Система представляє гібридний застосунок, який дозволяє проходити тестування по різним дисциплінам, завдання яких генеруються програмно з використання понятійно-тезисної мережі. Після закінчення оцінювання знань користувачеві виводиться результат. Особливістю застосунку є те, що його розробка була виконана за допомогою фреймворку Flutter.

ФУНКЦІОНАЛЬНЕ ПРИЗНАЧЕННЯ

Розроблений застосунок надає можливість користувачам проходити тестування по бажаній дисципліні, з отримання результатів тестування. Серед основного функціоналу можна виділити:

— Можливість введення імені і групи користувача.

— Отримання списку доступних дисциплін, по яким можна пройти тестування

— Вибір одного з рівнів складності

— Реалізація декількох типів тестових запитань

— Отримання результатів тестування

ОПИС ЛОГІЧНОЇ СТРУКТУРИ

Система складається з двох основних частин: клієнтської і серверної.

Серверна частина включає в себе функціональні можливості створення тестових завдань з використанням понятійно-тезисних мереж, які базуються на навчальному матеріалі по різним дисциплінам. Сервер має доступ до бази даних для збереження результатів роботи системи. Сервер відповідає за обробку запитів клієнтської частини і надає усі необхідні дані для її роботи.

Клієнтська частина відповідає за надання користувачам зручного інтерфейсу для роботи з системою. Вона поділена на 3 логічних шари:

— Шар UI компонентів

— Шар BLoC

— Шар репозиторіїв

ВИКОРИСТОВУВАНІ ТЕХНІЧНІ ЗАСОБИ

Для використання веб-застосунку користувач повинен мати персональний комп’ютер або мобільний пристрій з браузером, що підтримує JavaScript, а також підключення до мережі Інтернет.

Для роботи мобільного застосунку пристрій користувача має підтримувати операційну систему Android 4.1.x та новіші або IOS 8 та новіші.

ВИКЛИК І ЗАВАНТАЖЕННЯ

Веб застосунок не потребує інсталяції і доступна на сайті за посиланням http://test.semantic-portal.net/

ВХІДНІ ДАНІ

Вхідна інформація для додатку:

а) дані про користувача (ім’я і група)

б) відповіді користувача на тестові запитання.

ВИХІДНІ ДАНІ

Вихідна інформація додатку:

а) Результат тестування.

б) Перелік корисних посилань по темі питань, які зустрічались в тестуванні.

ДОДАТОК Г

Інструментальні засоби для тестування знань по різним дисциплінам з програмною генерацією питань

Апробації, публікації (Копії)

УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ТІ62628_20Б 4

Аркушів 2

Київ 2020

УДК 004.02

Студент 4 курсу, гр. ТІ-62 Заїчко О.П.
Доц., к.т.н. Гагарін О.О.

ЗАСОБИ ПІДВИЩЕННЯ ЯКОСТІ НАВЧАЛЬНОГО ПРОЦЕСУ НА ОСНОВІ ТЕХНОЛОГІЙ КОМП'ЮТЕРНОГО ТЕСТУВАННЯ

Система освіти може ефективно функціонувати, коли покращуються якісні показники роботи й оптимізуються кількісні показники. Для таких систем життєво необхідні «лінійки» об’єктивних показників, що впливають на якість і адекватно характеризують процес навчання.

В системі вищої освіти такими формальними, легко вимірюваними і контрольованими показниками є кваліфікаційний склад викладачів, площа учбових аудиторій, забезпеченість навчальними посібниками і т.д. Однак на якість самої освіти всі ці показники мають опосередкований вплив. Власне, якісні показники процесу освіти (оцінка знань студентів) важко формалізуються і складно оцінюються. Оцінка знань студентів за допомогою традиційного підходу оцінювання знань (контрольні заходи) не є об'єктивним і достовірним методом, не відповідає вимозі єдності вимірювань і не дозволяє будувати на її основі ефективну систему управління якістю освіти.

Тому в сьогодні стрімко розробляються і широко використовуються методи і технології комп'ютерного тестування (КТ). Традиційно прийнято звертати увагу на такі переваги КТ: масовість (можливість охоплення контролем великої кількості тестованих за певний проміжок часу), об'єктивність (виключення фактору суб'єктивного підходу з боку екзаменатора), оперативність і технологічність (можливість використання машинної обробки і представлення результатів тестування), порівнянність результатів і керованість (використання результатів тестування для виявлення типових помилок, облік яких дозволяє своєчасно скоригувати процес засвоєння навчального матеріалу) [1].

Існуючі системи КТ в більшості будуються на основі банків тестових завдань (БТЗ). Традиційно формування БТЗ є завданням авторів навчальних курсів і складно формалізується. У роботі пропонується формалізувати процес складання тестових завдань на базі семантичного аналізу навчальних матеріалів з виявленням базових понять і категорій їх взаємозв'язків [2]. Тоді, побудова шаблонів традиційних тестових завдань закритого типу на основі виявлених понять і їх зв'язків можливо алгоритмізувати.

Такий підхід реалізовано, побудована бета версія програмного комплексу, який проходить дослідну експлуатацію на сайтах: http://www.setlab.net, http://www.znannya.org та http://semantic-portal.net/

Перелік посилань:

1. Автоматизоване тестування в навчальному процесі [Електронний ресурс] – Режим доступу:
https://uk.wikipedia.org/wiki/Автоматизоване_тестування_в_навчальному_процесі
2. Титенко, С. В. Генерація тестових завдань у системі дистанційного навчання на основі моделі формалізації дидактичного тексту / С. В. Титенко // Наукові вісті НТУУ "КПІ". – 2009. – № 1(63). – С. 47–57.